Dieses Skript ist im Aufbau. Die Bezeichnungen können sich, wie auch der Inhalt, jederzeit ändern.

Kryptographie II

Dr. Sachar Paulus[1]

9. Februar 1998

[1]Skript wurde erstellt unter Mitwirkung von Lars Schuster (schuster@rbg.informatik.tu-darmstadt.de)

Kapitel 1

Klassische Kryptosysteme

Dieses Kapitel beschäftigt sich mit den klassischen Kryptosystemen. Das sind diejenigen Systeme, die bis zum zweiten Weltkrieg eingesetzt wurden. Sie sind klar zu trennen von den heutigen symmetrischen Kryptosystemen, da sie deren Sicherheitsstandard nicht erfüllen können. Zuerst eine allgemeine Definition:

1.1. Definition *Ein **Kryptosystem** ist ein 5-Tupel* $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ *von endlichen Mengen, wobei für alle* $K \in \mathcal{K}$ *es ein Paar* $(e_K, d_K) \in \mathcal{E} \times \mathcal{D}$ *gibt mit:*

- $e_K : \mathcal{P} \longrightarrow \mathcal{C}$,
- $d_K : \mathcal{C} \longrightarrow \mathcal{P}$ *und*
- *für alle* $X \in \mathcal{P}$ *gilt* $d_K(e_K(X)) = X$.

$\mathcal{P}$ ist die Menge der **Klartexte** (Messages, Plaintexte), $\mathcal{C}$ ist die Menge der **Chiffretexte**, $\mathcal{K}$ ist die Menge der **möglichen Schlüssel** (Schlüsselraum), $\mathcal{E}$ ist die Menge der **Verschlüsselungsfunktionen** und $\mathcal{D}$ ist die Menge der **Entschlüsselungsfunktionen**. Wir sagen, das Paar (e_K, d_K) **ist durch** K **parametrisiert**.

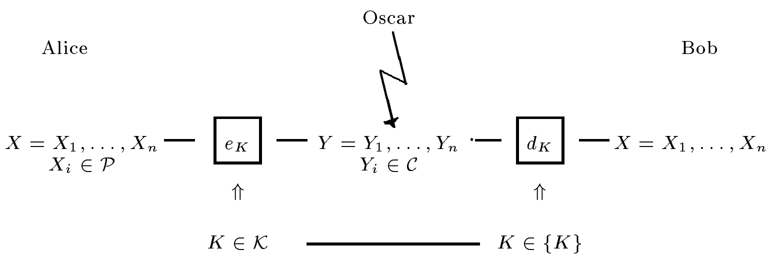


Tabelle 1.1: Schema eines klassischen Kryptosystems

Falls $\mathcal{P} = \mathcal{C}$, dann gilt $\mathcal{E}, \mathcal{D} \subset \Sigma_{|\mathcal{P}|}$. D.h. alle Elemente von $\mathcal{E}$ und $\mathcal{D}$ können als **Permutationen** auf der Menge der Klartexte aufgefaßt werden. Für eine endliche Menge M bezeichnet $|M|$ die Anzahl der Elemente von M.

Wir identifizieren das Alphabet mit $\mathbb{Z}/26\mathbb{Z} = \mathbb{Z}_{26}$ wobei $A \leftrightarrow 0$, $B \leftrightarrow 1$, usw. bis $Z \leftrightarrow 25$. $\mathbb{Z}_{26}$ ist ein **Ring**, d.h. $(\mathbb{Z}_{26}, +)$ ist eine **Gruppe** und die Multiplikation verhält sich **distributiv** zur

Addition ($+, \cdot$ werden „modulo 26“ ausgeführt). Allgemeiner betrachten wir Alphabete mit m Elementen in dieser Weise wie den Ring $\mathbb{Z}_m$.

Anforderungen an ein Kryptosystem:

- Effizienz: Die Funktionen e_K und d_K sollten effizient zu berechnen sein.
- Sicherheit: Ein Lauscher (Oskar) sollte nicht in der Lage sein, K oder X nur aus Y bestimmen zu können.

1.1 Die Verschiebungschiffe

Sei $m \in \mathbb{N}$. Die *Verschiebungsschiffre* ist ein Kryptosystem $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{K} = \mathcal{C} = \mathbb{Z}_m$,
2. für alle $K \in \mathcal{K}$ ist $e_K(X) = X + K \bmod m$ und $d_K(Y) = Y - K \bmod m$.

Wir müssen nachweisen, daß für alle $K \in \mathcal{P}$ und für alle $X \in \mathcal{P}$ gilt $d_K(e_K(X)) = X$. Nun ist

$$d_K(e_K(X)) = d_K(X + K) = (X + K) - K = X.$$

Bekanntestes Beispiel ist die Cäsar Chiffre, für welche $m = 26$ und $K = 3$. Man kann sich die Verschiebungschiffre vorstellen wie zwei runde Scheiben mit verschiedenem Radius, die an ihrem Mittelpunkt befestigt sind und mit dem Alphabet im Kreis beschrieben sind.

1.2. Beispiel Wir wählen $m = 26$, $K = 5$. Hier die Verschlüsselung:

H	U	N	G	E	R	H	A	B	I	C	H
7	20	13	6	4	17	7	0	1	8	2	7
12	0	18	11	9	22	12	5	6	13	7	12
M	A	S	L	J	W	M	F	G	N	H	M

Mit etwas Sprachgefühl kann man durch sukzessives Probieren der m möglichen Schlüssel nach im Schnitt $\lceil m/2 \rceil$ Versuchen die Verschiebungsschiffre knacken.

1.2 Die Substitutionschiffre

Sei $m \in \mathbb{N}$. Die *Substitutionschiffre* ist ein Kryptosystem $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_m$,
2. $\mathcal{K} = \Sigma_m$,
3. für alle $\pi \in \mathcal{K}$ ist $e_\pi(X) = \pi(X)$ und $d_\pi(Y) = \pi^{-1}(Y)$.

Wir müssen nachweisen, daß für alle $\pi \in \mathcal{P}$ und für alle $X \in \mathcal{P}$ gilt $d_\pi(e_\pi(X)) = X$. Nun ist

$$d_\pi(e_\pi(X)) = d_\pi(\pi(X)) = \pi^{-1}(\pi(X)) = X.$$

Das Verschlüsseln mit der Substitutionsschiffre entspricht einem Nachschlagen in einer Tabelle und Ersetzen durch einen anderen Buchstaben.

1.3. Beispiel $m = 26$, π gegeben durch folgende Tabelle:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S T...
Q	W	E	R	T	Z	U	I	O	P	A	S	D	F	G	H	J	K	L Y...

Dann ist die Verschlüsselung von folgendem Text so:

G	L	E	I	C	H	G	I	B	T	S	E	S	S	E	N
U	S	T	O	E	I	U	O	W	Y	L	T	L	L	T	F

Die Größe der Tabelle bestimmt die Effizienz, da man für jedes Zeichen in der Tabelle nachsehen muß.

Sukzessives Probieren der Schlüssel ist bei wachsender Tabelle schnell unmöglich, denn es gibt $m!$ viele Permutationen. Für $m = 26$ sind das ungefähr 10^{26} Versuche. Trotzdem ist die Substitutionsschiffre nicht sicher. Siehe später.

1.3 Die affine Chiffre

Sei $m \in \mathbb{N}$. Die *affine Chiffre* ist ein Kryptosystem $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_m$,
2. $\mathcal{K} = \{(a, b) \in \mathbb{Z}_m^2 : \text{ggT}(a, m) = 1\}$,
3. für alle $(a, b) \in \mathcal{K}$ ist $e_{(a,b)}(X) = a \cdot X + b$ und $d_{a,b}(Y) = a^{-1}(Y - b)$.

Es gibt a^{-1} genau dann, wenn $\text{ggT}(a, m) = 1$. Wir müssen nachweisen, daß für alle $(a, b) \in \mathcal{P}$ und für alle $X \in \mathcal{P}$ gilt $d_{(a,b)}(e_{(a,b)}(X)) = X$. Nun ist

$$d_{(a,b)}(e_{(a,b)}(X)) = d_{(a,b)}(a \cdot X + b) = a^{-1}((a \cdot X + b) - b) = X.$$

Die affine Chiffre nutzt die Ringstruktur vom $\mathbb{Z}_m$ aus.

1.4. Beispiel $m = 26$, $K = (7, 3)$. Nun ist $7 \cdot 15 \equiv 1 \bmod 26$, also ist $7^{-1} = 15$ in $\mathbb{Z}_{26}$. Verschlüsselt wird wie folgt:

$$e_{(a,b)}(X) = 7 \cdot X + 3 \bmod 26$$

Entschlüsseln geht dann so:

$$d_{(a,b)}(Y) = 15 \cdot (Y - 3) \bmod 26.$$

B	A	L	D
1	0	11	3
10	3	2	24
K	D	C	Y

1.4 Die Vigenère Chiffre (16. Jhd)

Seien $m, n \in \mathbb{N}$. Die *Vigenère Chiffre* ist ein Kryptosystem $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_m^n$ und
2. für alle $K = (k_1, \ldots, k_n) \in \mathcal{K}$ ist $e_K(x_1, \ldots, x_n) = (x_1 + k_1, \ldots, x_n + k_n)$ und $d_K(y_1, \ldots, y_n) = (y_1 - k_n, \ldots, y_n - k_n)$.

Eine solche Chiffre, welche mehrere Zeichen des Alphabets gleichzeitig verschlüsselt, heißt **poly-alphabetisch**, die bisher betrachteten hingegen **monoalphabetisch**.

1.5. Beispiel $m = 26$, $n = 5$, $K = MENSA \sim (12, 4, 13, 18, 0)$.

	D	O	S	C	H	M	E	C	K	T	S	G	U	D	D
	3	14	18	2	7	12	4	2	10	19	18	6	20	3	3
+	12	4	13	18	0	12	4	13	18	0	12	4	13	18	0
	15	18	5	20	7	24	8	15	2	19	4	10	7	21	3
	P	S	F	U	H	Y	I	P	C	T	E	K	H	V	D

Die Anzahl der möglichen Schlüssel beträgt m^n. Für $n = 5$, $m = 26$: $26^5 \approx 10^7$.

1.5 Die Hill Chiffre (1929)

Seien $m, n \in \mathbb{N}$. Die *Hill Chiffre* ist ein Kryptosystem $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_m^n$,
2. $\mathcal{K} = \mathcal{E} = \mathcal{D} = \{A \in \mathbb{Z}^{n \times n} : \mathrm{ggT}(\det A, m) = 1\}$ und
3. für alle $A \in \mathcal{K}$ ist $e_A(X) = X \cdot A$ und $d_A(Y) = y \cdot A^{-1}$

1.6. Beispiel $m = 26$, $n = 2$, $A = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$, $A^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$.

M	I	T	T	A	G	S	T	I	S	C	H
12	8	19	19	0	6	18	19	8	18	2	7

$$(12, 8) \cdot \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (12 \cdot 11 + 8 \cdot 3, 12 \cdot 8 + 8 \cdot 7) = (156, 152) = (0, 22) \bmod 26$$

0	22	6	25	18	16	21	17	12	18	9	13
A	W	G	Z	S	Q	V	R	M	S	J	N

Aufgabe: Wie würde eine mehrdimensionale Verallgemeinerung der affinen Chiffre aussehen? Was wäre der Unterschied zur Hill Chiffre?

1.6 Die Permutationschiffre

Seien $m, n \in \mathbb{N}$. Die *Permutationschiffre* ist ein Kryptosystem $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_m^n$,
2. $\mathcal{K} = \Sigma_n$ und
3. für alle $\pi \in \mathcal{K}$ ist $e_\pi(x_1, \ldots, x_n) = (x_{\pi(1)}, \ldots, x_{\pi(n)})$ und $d_\pi(y_1, \ldots, y_n) = (y_{\pi^{-1}(1)}, \ldots, y_{\pi^{-1}(n)})$.

1.7. Beispiel $m = 26$, $n = 5$, $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 5 & 3 & 2 \end{pmatrix}$.

M	E	I	N	M	A	G	E	N	K	N	U	R	R	T	S	C	H	O	N
		$\downarrow$	π				$\downarrow$	π				$\downarrow$	π				$\downarrow$	π	
N	M	M	I	E	N	A	K	E	G	R	N	T	U	N	O	S	N	H	C

1.8. Proposition *Die Permutationschiffre ist ein Spezialfall der Hill Chiffre.*

Beweis: Übung. ■

1.7 Stromchiffren

Bisher haben wir **Blockchiffren** betrachtet:

$$sY = Y_1, Y_2, \ldots = e_K(X_1), e_K(X_2), \ldots$$

Dabei wird derselbe Schlüssel K für jeden Block von Klartextzeichen verwandt. Wir modifizieren nun dieses Verfahren und erzeugen einen Schlüssel-„Strom“ $Z = Z_1, Z_2 \ldots$ und verschlüsseln

$$Y = Y_1, Y_2, \ldots = e_{Z_1}(X_1), e_{Z_2}(X_2), \ldots$$

1.9. Definition *Eine* Stromchiffre *ist ein 7-Tupel* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{Z}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ *mit*

1. $\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{Z}, \mathcal{E}, \mathcal{D}$ *endliche Mengen,*
2. $\mathcal{F} = (f_1, f_2, \ldots)$ *Folge von Funktionen* $f_i : \mathcal{K} \times \mathcal{P}^{i-1} \to \mathcal{Z}$,
3. *für alle* Z *in* $\mathcal{Z}$ *gibt es ein Paar* $(e_Z, d_Z) \in \mathcal{E} \times \mathcal{D}$ *mit:*
   - $e_Z : \mathcal{P} \longrightarrow \mathcal{C}$,
   - $d_Z : \mathcal{C} \longrightarrow \mathcal{P}$ *und*
   - *für alle* $X \in \mathcal{P}$ *gilt* $d_Z(e_Z(X)) = X$.

$\mathcal{Z}$ ist das **Schlüsselstromalphabet** und $\mathcal{F}$ ist der **Schlüsselstromerzeuger**.

1.10. Proposition *Jede Blockchiffre ist eine Stromchiffre.*

Beweis: Sei $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ eine Blockchiffre. Sei nun $\mathcal{Z} = \mathcal{K}$ und $\mathcal{F} = (f_1, f_2, \ldots, ...)$ mit $f_i : \mathcal{K} \times \mathcal{P}^{i-1} \to \mathcal{Z}$ und $f_i(K, X_1, \ldots, X_i - 1) = K$. Dann ist $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{Z}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ eine Stromchiffre, welche den gleichen Chiffretext erzeugt wie die gegebene Blockchiffre. ■

1.11. Definition *1. Eine Stromchiffre heißt* ***synchron****, wenn für jedes $f_i \in \mathcal{F}$ z_i nicht vom Klartext abhängt.*

2. Eine Stromchiffre heißt ***periodisch****, falls für alle Schlüssel $K \in \mathcal{K}$ es $i_0, d \in \mathbb{N}$ gibt, so daß für alle $i > i_0$ $Z_i = Z_{i+d}$ gilt. In diesem Falle heißt d* ***Periode****.*

1.12. Beispiel Die Vigenère Chiffre mit Schlüsselwortlänge n ist eine synchrone periodische Stromchiffre mit Periode n (s. Aufgabe I/2).

Ab nun betrachten wir Stromchiffren mit binärem Alphabet, d.h. $\mathcal{P} = \mathcal{C} = \mathcal{Z} = \mathbb{Z}_2$ und mit XOR-Verschlüsselung, d.h. $e_Z(X) = X \oplus Z = X + Z \bmod 2$ und $d_Z(Y) = Y \oplus Z = Y + Z \bmod 2$.

Sei weiterhin $K = (k_1, \ldots, k_n, c_1, \ldots, c_n)$ und

$$z_i = \begin{cases} k_i & \text{für } 1 \leq i \leq n, \\ \bigoplus_{j=1}^{n} c_j \times z_{n-j} & \text{sonst.} \end{cases}$$

Auf diese Weise kann man Stromchiffren mit langer Periode erzeugen. Solche Stromchiffren haben **lineare Rekurrenz** vom Grad n.

1.13. Beispiel $m = 2$, $n = 4$, $K = 10001100$. Dann ist die Periode 15 und der Schlüsselstrom ist $100010011010111\ldots$.

Binäre Stromchiffren mit linearer Rekurrenz können durch die Realisierung der linearen Rekurrenz mit XOR-Gattern effizient in Hardware realisiert werden (**Linear Feeback Shift Register**, LFSR).

Im letzten Beispiel:

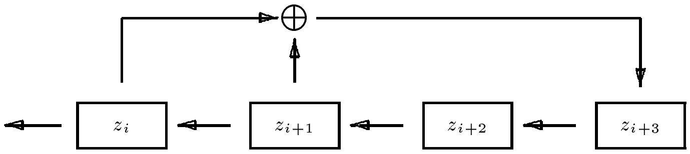


Tabelle 1.2: Linear Feedback Shift Register

1.8 Die Autokey Chiffre

Sei $m \in \mathbb{N}$. Die *Autokey Chiffre* ist eine Stromchiffre $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{Z}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ mit

1. $\mathcal{P} = \mathcal{K} = \mathcal{C} = \mathcal{Z} = \mathbb{Z}_m$,
2. $Z_1 = K$, $Z_i = X_{i-1}$ für $i > 1$ und

3. für alle $Z \in \mathcal{Z}$ ist $e_Z(X) = X + Z$ und $d_Z(Y) = Y - Z$.

1.14. Beispiel $m = 26$, $K = 1$.

	S	C	H	L	U	S	S	J	E	T	Z	T
	19	2	7	11	21	19	19	9	4	20	25	20
+	1	19	2	7	11	21	19	19	9	4	20	25
	20	21	9	19	6	14	12	2	13	24	19	19
	T	U	J	S	G	O	M	B	N	Y	S	S

Die Autokey Chiffre ist eine asynchrone Stromchiffre. Es gibt auch die Variante mit $Z_i = Y_{i-1}$ für $i > 1$. Eine einzelne Fehlentschlüsselung führt dabei dazu, daß man den restlichen Text nicht mehr verstehen kann.

Kapitel 2

Kryptoanalyse

Wir gehen in diesem Kapitel von der allgemeinen Annahme aus, daß Oskar das Kryptosystem kennt (*Kerckhoff's Prinzip*). Wir beschäftigen uns nicht mit Methoden, das gerade verwendete Kryptosystem zu identifizieren.

Die Angriffe werden nach aufsteigender Stärke in 4 Klassen eingeteilt:

- **Ciphertext only**: Nur der abgefangene Chiffretext steht zur Verfügung.
- **Known plaintext**: Bekannte Klartext-Chiffretextpaare stehen zur Verfügung (z.B. „mit freundlichen Grüßen“).
- **Chosen plaintext**: Oskar hat (direkten oder indirekten) Zugang zur Verschlüsselungsfunktion.
- **Chosen ciphertext**: Oskar hat (direkten oder indirekten) temporären Zugang zur Entschlüsselungsfunktion. Hier gilt das Interesse der Bestimmung des Schlüssels.

Die Methoden der Kryptoanalyse lassen sich wie folgt gliedern:

1. Vollstandiges Durchsuchen des Schlüsseltextraumes und Ausprobieren aller Schlüssel.
2. Trial and Error. „Intelligentes“ Probieren indem man gewisse Eigenschaften des Kryptosystems ausnutzt ohne jedoch ein systematisches Verfahren anzuwenden.
3. Statistische Methoden. Hierbei werden statische Eigenschaften der Klartextsprache ausgenutzt. Z.B. häufig auftretende Zeichenfolgen.
4. Strukturanalyse. Konkrete mathematische Analyse des Verfahrens mit dem Ziel, ein maßgeschneidertes „Knack“ -Verfahren zu entwickeln.

Wir werden 3. und 4. näher betrachten.

A	B	C	D	E	F	G	H	I	J	K	L	M
82	15	28	43	127	22	20	61	70	2	8	40	24
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
67	75	19	1	60	63	91	28	10	23	1	20	1

Tabelle 2.1: Häufigkeit einzelner Buchstaben in der englischen Sprache auf 1000 Zeichen

2.1 Statistische Methoden

Wir benutzen statistische Aussagen über die verwendete Sprache, und zwar über die Häufigkeit einzelner Buchstaben, von Buchstabenfolgen der Länge 2 (*Bigramme*) sowie der Länge 3 (*Trigramme*). Für die englische Sprache siehe Tabelle 2.1.

Die 10 häufigsten Bigramme in der englischen Sprache sind TH, HE, EN, IR, AN, RE, ED, ON, ES, ST. Die 5 häufigsten Trigramme sind THE, ING, AND, HER, ERE.

Bezeichne im Folgenden für ein Zeichen oder eine Zeichenkette A die Wahrscheinlichkeit ihres Auftretens in einem unendlich langen englischsprachigen Text mit p_A.

Einfache statistische Methoden können eingesetzt werden, um monoalphabetische Chiffren wie die Substitutionschiffre und die Affine Chiffre zu brechen.

Beispiele: S. Übung.

2.1.1 Der Kasiski-Test (1863)

Der Kasiski-Test wird eingesetzt, um die Schlüsselwortlänge bei einer Vigenère Chiffre zu bestimmen. Sei n die zu bestimmende Schlüsselwortlänge.

Der Test beruht auf der folgenden Beobachtung: zwei identische Buchstabenfolgen werden auf das gleiche Chiffretextstück abgebildet, wenn sie $k \cdot n$ für $k \in \mathbb{N}$ Zeichen auseinander liegen.

Es ergibt sich sofort folgende probabilistische Methode: Suche nach Paaren von identischen Chiffretextsegmenten der Länge ≥ 3 und betrachte deren Abstände. Treten mehrere identische Segmente auf, so könnte n den ggT der zugehörigen Abstände teilen.

2.1.2 Der Koinzidenzindex (1920)

Der Koinzidenzindex ist eine weitere Methode, um die Schlüsselwortlänge einer Vigenère Chiffre zu bestimmen. Sie wurde von W. Friedman entwickelt und ist im Gegensatz zum Kasiski-Test ein deterministisches Verfahren.

2.1. Definition *Sei $\mathbb{Z}_m$ ein endliches Alphabet und sei $X = X_1 \dots X_l$ ein Wort über $\mathbb{Z}_m$. Der* ***Koinzidenzindex*** *$I_C(X)$ von X ist die Wahrscheinlichkeit, daß zwei zufällig gewählte Zeichen X_i und X_j von X mit $i \neq j$ gleich sind.*

Der Koinzidenzindex kann wie folgt berechnet werden:

2.2. Lemma *Sei $\mathbb{Z}_m$ ein endliches Alphabet und sei $X = X_1 \ldots X_l$ ein Wort über $\mathbb{Z}_m$ mit $l > 0$. Bezeichne $h_X(A) = \#\{X_i | i = 1, \ldots, l \quad X_i = A\}$. Dann ist*

$$I_C(X) = \frac{1}{l(l-1)} \sum_{A \in \mathbb{Z}_m} h_X(A)(h_X(A) - 1).$$

Beweis: Sei $A \in \mathbb{Z}_m$ mit $h(A) \geq 1$. Die Wahrscheinlichkeit, daß ein zufällig gewähltes Zeichen von X gleich A ist, beträgt $h_X(A)/l$. Die Wahrscheinlichkeit, daß ein weiteres gewähltes Zeichen von X gleich A ist, beträgt $\frac{h_X(A)-1}{l-1}$. Somit ist die Wahrscheinlichkeit, daß zwei gewählte Zeichen gleich A sind, $\frac{h_X(a)/h_X(a)-1)}{l(l-1)}$. Kommt A in X nicht vor, dann ist $h_X(A) = 0$ und so wird auch dieser Fall erfaßt. Summation über alle Zeichen von $\mathbb{Z}_m$ liefert schließlich die Aussage. ■

Wir vergleichen nun den Koinzidenzindex eines gegebenen Chiffretextes mit dem „Sollwert" eines englischsprachigen Textes. Sei X_∞ ein unendlich langer Text der englischen Sprache. Dann ist die Wahrscheinlichkeit, daß zwei beliebig gewählte Zeichen gleich A sind, p_A^2. Also ist

$$I_C(X_\infty) = \sum_{A \in \mathbb{Z}_m} p_A^2.$$

Für die englische Sprache ist z.B. $I_C(X_\infty) \approx 0.065$. Also erwarten wir für einen englischsprachigen Text X, daß $0.060 \leq I_C(X) \leq 0.070$.

Haben wir einen mit einer monoalphabetischen Chiffre verschlüsselten Text vorliegen, so bleibt der Konzidenzindex unverändert. Wurde eine Vigenère Chiffre verwendet, so gilt dies i.a. nicht mehr. Es gilt aber folgendes: Wenn die Blocklänge der Chiffre n ist und man die n Texte betrachtet, die entstehen, wenn man jeweils nur jedes n-te Zeichen nimmt, dann kann man wieder den obigen Durchschnittswert erwarten.

Es ergibt sich also folgendes Verfahren:

Sei $Y = Y_1, \ldots, Y_l$ ein Vigenère Chiffretext. Sei im folgenden $n = 1, 2, \ldots$ jeweils fest. Für jedes $1 \leq i \leq n-1$ sei $Y^{(i,n)} = Y_i Y_{i+n} Y_{i+2n} \ldots$; berechne $I_C(Y^{i,n})$ für jedes i. Sind diese Koinzidenzindizes in der Nähe von 0.065 für alle i, so ist n mit großer Wahrscheinlichkeit die Blocklänge der Chiffre.

2.1.3 Der simultane Konzidenzindex

Der simultane Koinzidenzindex ist eine Verfeinerung des Koinzidenzindices und wird verwendet, um den Schlüssel einer Vigenère Chiffre zu bestimmen.

2.3. Definition *Sei $\mathbb{Z}_m$ ein endliches Alphabet und seien $X = X_1 \ldots X_l$ und $Y = Y_1 \ldots, Y_l'$ Worte über $\mathbb{Z}_m$. Der* ***simultane Koinzidenzindex*** *$I_C(X,Y)$ von X und Y ist die Wahrscheinlichkeit, daß ein zufällig gewähltes Zeichen X_i von X und ein zufällig gewähltes Zeichen Y_j von Y gleich sind.*

Der simultane Koinzidenzindex von X und Y kann wie folgt berechnet werden:

2.4. Lemma *Sei $\mathbb{Z}_m$ ein endliches Alphabet und seien $X = X_1 \ldots X_l$ und $Y = Y_1, \ldots, Y_l'$ Worte über $\mathbb{Z}_m$ mit $l, l' > 0$. Bezeichne $h_X(A) = \#\{X_i | i = 1, \ldots, l \quad X_i = A\}$. Dann ist*

$$I_C(X,Y) = \frac{1}{l \cdot l'} \sum_{A \in \mathbb{Z}_m} h_X(A) \cdot h_Y(A).$$

Beweis: Analog zu obigem Lemma. ■

Sei n die Blocklänge, sei $K = K_1 \dots K_n$ der verwendete Schlüssel und bezeichne

$$Y^i = Y_i Y_{i+n} Y_{i+2n} \dots$$

. Wir vergleichen nun Y^i und Y^j und bestimmen daraus den „Unterschied“ zwischen K_i und K_j. Um dies zu erreichen, vergleichen wir wie oben den simultanen Kongruenzindex unserer gegebenen Strings mit hypothetischen unendlich langen Strings.

X und Y entsprechend anordnen!!

2.5. Lemma *Sei X_∞ ein unendlich langer String englischer Sprache, verschlüsselt zu Y_∞. Dann gilt für den simultanen Konzidenzindex von Y^i_∞ und Y^j_∞*

$$I_C(Y^i_\infty, Y^j_\infty) \approx 0.065 \quad \textit{wenn} \quad K_i = K_j.$$

Beweis: Wir betrachten die Wahrscheinlichkeit, daß ein zufällig gewähltes Zeichen aus Y^i_∞ und ein zufällig gewähltes Zeichen aus Y^j_∞ gleich sind.

$$\begin{aligned} p(Y^i_l = Y^j_m) &= p(X_{l\cdot n+i} + K_i = X_{m\cdot n+j} + K_j) \\ &= p(X_{l\cdot n+i} = X_{m\cdot n+j} + (K_j - K_i)) \end{aligned}$$

Also ist $I_C(Y^i_\infty, Y^j_\infty) = I_C(X_\infty)$, wenn $K_i = K_j$. Daraus folgt die Behauptung. ■

Man bemerke, daß der Umkehrschluß nicht unbedingt gilt. Trotzdem gibt uns dieses Lemma eine Möglichkeit, die Unterschiede zwischen den verschiedenen Schlüsselzeichen zu bestimmen.

Dies funktioniert wie folgt: wir halten Y^i fest und verschlüsseln Y^j “künstlich” mit allen möglichen G, d.h. wir addieren G auf jedes einzelne Zeichen von Y^j. Das Ergebnis dieser Berechnung nennen wir $Y^{j,G}$. Nun berechnen wir den simultanen Koinzidenzindex; ist dieser in der Nähe von 0.065, dann haben wir $K_i - K_j = G$. Wir bekommen schließlich ein lineares Gleichungssystem zwischen den Schlüsselzeichen K_i und K_j. Eventuell müssen wir manche der herangezogenen Werte der I_C weglassen oder neue hinzunehmen, da es vorkommen kann, daß $I_C(Y^i, Y^{j,G}) \approx 0.065$, aber $K_i \neq K_j$ gilt.

2.2 Strukturanalyse

Wir stellen als Beispiel eine known-plaintext-Attacke vor, mit der man die Hill-Chiffre brechen kann. Wir werden im (interessanteren) Fall von DES ein ausführlicheres Beispiel für eine Strukturanalyse - kombiniert mit statistischen Aussagen - kennenlernen.

Der Angriff beruht nicht auf statistischer Auswertung des Chiffretextes; für diesen Angriff benötigt man die Blockgröße n und mindestens n verschiedene Paare von Klartexten und Chiffretexten. Ziel ist hierbei die Bestimmung des Schlüssels, also der Transformationsmatrix A. Bezeichne X die Matrix über $\mathbb{Z}_m$, die entsteht, wenn man die Klartexte als Zeilen auffaßt; analog für Y. Es entsteht eine Matrixgleichung

$$Y = X \cdot K,$$

welche lösbar ist, wenn X invertierbar ist. In diesem Fall gilt $K = X^{-1}Y$. Ist X nicht lösbar, müssen weitere Klartext-Chiffretextpaare bestimmt werden.

Ist die Blocklänge n nicht bekannt, so probiere man dieses Verfahren sukzessive für $n = 1, 2, \dots$ bis K sinnvoll ist, d.h. auch weitere Klartexte in Schlüsseltexte überführt.

Kapitel 3

Informationstheorie

In diesem Kapitel werden wir die theoretische Sicherheit von Kryptosystemen betrachten, d.h. wir nehmen an, daß uns unendlich viel Rechenleistung zur Verfügung steht. Diese in der Praxis unrealistische Voraussetzung hat jedoch einige beachtenswerte Konsequenzen.

3.1 Perfekte Sicherheit

Seien X und Y Zufallsvariablen. Wir bezeichnen die Wahrscheinlichkeit, daß X den Wert x annimmt, mit $p(x)$. Wir bezeichnen die gemeinsame Wahrscheinlichkeit, daß X den Wert x und Y den Wert y annimmt, mit $p(x, y)$. Wir bezeichnen die bedingte Wahrscheinlichkeit, daß X den Wert x annimmt, sofern Y den Wert y annimmt, mit $p(x \mid y)$. X und Y heißen unabhängig, falls $p(x, y) = p(x) \cdot p(y)$ für alle möglichen Werte x und y.

Es gilt:

- $p(x, y) = p(x \mid y) \cdot p(y) = p(y \mid x) \ldots p(x)$ und
- X und Y sind genau dann unabhängig, wenn für alle x, y gilt $p(x \mid y) = p(x)$.

Ab sofort bezeichnen wir die Zeichen des Klartextes mit x, y usw. Wir gehen davon aus, daß ein bestimmter Schlüssel nur zur Verschlüsselung von einem Klartext(-zeichen) verwendet wird. Wir gehen weiterhin davon aus, daß es Wahrscheinlichkeitsverteilungen für das Auftreten der Klartextzeichen, Schlüssel und Chiffretextzeichen gibt. Wir bezeichnen dann die Wahrscheinlichkeit für das Auftreten

- eines Klartextzeichens x mit $p_P(x)$,
- eines Schlüssels k mit $p_K(k)$ und
- eines Chiffretextzeichens c mit $p_C(c)$.

3.1. Proposition *Sei $k \in \mathcal{K}$. Bezeichne*

$$C(k) = \{e_k(x) \mid x \in \mathcal{P}\}$$

die Menge der Bilder unter e_k. Es gelten die folgenden Identitäten:

- $p_C(c) = \sum\limits_{\{k \mid c \in C(k)\}} p_K(k) \cdot p_P(d_k(c))$,
- $p_C(c \mid x) = \sum\limits_{\{k \mid x = d_k(c)\}} p_K(k)$ *und*
- $p_P(x \mid c) = \dfrac{p_P(x) \sum\limits_{\{k \mid x = d_k(c)\}} p_K(K)}{\sum\limits_{\{k \mid c \in C(k)\}} p_K(k) \cdot p_{\mathcal{P}}(d_K(c))}$

3.2. Definition *Ein Kryptosystem hat* **perfekte** *oder* **absolute Sicherheit***, wenn für alle $x \in \mathcal{P}$ und alle $c \in \mathcal{C}$ gilt: $p_P(x \mid c) = p_P(x)$.*

d.h. Oskar kann aus dem Chiffretext absolut keine Information über den Klartext gewinnen.

3.3. Satz *Werden die Schlüssel im Verschiebungschiffre alle mit gleicher Wahrscheinlichkeit benutzt und wird für jedes Klartextzeichen ein eigener Schlüssel benutzt, so hat diese Chiffre perfekte Sicherheit.*

Beweis: $\mathcal{P} = \mathcal{K} = \mathcal{C} = \mathbb{Z}_m$, $e_k(x) = x + k \bmod m$. Dann ist

$$\begin{aligned} p_C(c) &= \sum_{k \in \mathbb{Z}_m} p_K(k) \cdot p_P(d_k(c)) \\ &= \sum_{k \in \mathbb{Z}_m} \frac{1}{m} p_P(c-k) \\ &= \frac{1}{m} \sum_{k \in \mathbb{Z}_m} p_P(c-k) \\ &= \frac{1}{m} \sum_{y \in \mathbb{Z}_m} p_P(y) \\ &= \frac{1}{m} \end{aligned}$$

und andererseits $p_C(c \mid x) = p_K(c - x) = 1/m$, also haben wir

$$p_P(x \mid c) = \frac{p_P(x) \cdot p_C(c \mid x)}{p_C(c)} = \frac{p_P(x) \cdot 1/m}{1/m} = p_P(x).$$

■

Etwas allgemeiner können wir folgenden Satz formulieren:

3.4. Satz *Sei $S = (\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem.*

- *Hat S perfekte Sicherheit, dann gilt $|\mathcal{P}| \leq |\mathcal{C}| \leq |\mathcal{K}|$.*
- *Gilt $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$, so hat S genau dann perfekte Sicherheit, wenn*
  1. *für alle $k \in \mathcal{K}$ gilt $p_K(k) = 1/|\mathcal{K}|$ und*
  2. *für alle $x \in \mathcal{P}$ und für alle $c \in \mathcal{C}$ gibt es genau ein $k \in \mathcal{K}$ mit $e_k(x) = c$.*

Beweis: S. Übung. ■

3.5. Definition *Ein* One-Time-Pad *ist eine Vigenère Chiffre* $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ *mit* $\mathcal{P} = \mathcal{K} = \mathcal{C} = \mathbb{Z}_2^n$*, wobei* n *die Länge der zu übertragenden Nachricht ist.*

3.6. Satz *Werden die Schlüssel im One-Time-Pad mit gleicher Wahrscheinlichkeit gewählt, so hat das One-Time-Pad perfekte Sicherheit.*

Beweis: S. Übung. ■

Um einen n Bit langen Klartext zu verschlüsseln, benötigt man somit Schlüssel der Länge n. Weiterhin muß für jede Übertragung ein zufällig gewählter Schlüssel ausgetauscht werden. Dies stellt meist zu große Anforderungen an das Schlüsselmanagement. Das One-Time-Pad wurde schon 1917 von Vernam patentiert und ist sehr einfach zu implementieren. Es findet vorrangig im militärischen/diplomatischen Bereich Einsatz.

3.2 Entropie und Redundanz

Statt von Zufallsvariablen werden wir in diesem Abschnitt von Nachrichtenquellen sprechen. Durch diese Begriffsänderung wird die Meßbarkeit der übermittelten Information ersichtlich, was Inhalt dieses Abschnitts ist.

3.7. Definition *Es seien* $x_1, \ldots, x_n$ *die* n *Nachrichten einer Nachrichtenquelle* X*, die mit den Wahrscheinlichkeiten* $p(x_i) > 0$ *auftreten. Dann heißt*

$$H(X) := \sum_{i=1}^{n} p(x_i) \log \left(\frac{1}{p(x_i)} \right)$$

Entropie *von* X*.*

Die Bedingung $p(x_i) > 0$ ist nur für eine saubere Definition notwendig. In der Praxis betrachtet man dann nur die x_i, die tatsächlich auftreten. Die Entropie drückt anschaulich die durchschnittliche Anzahl von Bits aus, die benötigt werden, um eine Nachricht mit einem optimalen Code zu generieren. Ein solcher Code heißt **Huffman-Code**. Somit kann man den Informationsgehalt einer Nachricht quantisieren.

3.8. Beispiel $X = \{x_1, \ldots, x_7\}$, $p(x_1) = p(x_2) = 1/4$, $p(x_3) = p(x_4) = p(x_5) = 1/8$, $p(x_6) = p(x_7) = 1/16$. Dann ist $H(X) \approx 2,625$.

Sind alle Nachrichten gleichwahrscheinlich, dann gilt $H(X) = \log n$. $H(X)$ ist maximal, wenn p die Gleichverteilung ist, also gilt $H(X) \leq \log n$. $H(X)$ wird kleiner, je „vorhersagbarer“ die Nachrichten sind. Schließlich wird $H(X) = 0$ falls $p(x) = 1$ für ein x.

3.9. Definition *Seien X,Y zwei Nachrichtenquellen. Für jedes y von Y haben wir eine Quelle $X_{|y}$ und eine entsprechende Entropie*

$$H(X_{|y}) = \sum_{x \in X} p(x \mid y) \log \left(\frac{1}{p(x \mid y)} \right).$$

$H(X \mid Y) := \sum_{y \in Y} p(y) H(X_{|y})$ *ist die* **bedingte Entropie** *von X, gegeben Y.*

Die bedingte Entropie ist für uns das wichtigste Maß, denn damit können wir die Informationsmenge quantisieren, die wir durch Preisgabe von Teilinformationen erreichen können.

3.10. Satz *Es gelten die folgenden Beziehungen:*

- $H(X \mid Y) \leq H(X)$,
- $H(X \mid Y) = H(X) \Leftrightarrow X$ *und* Y *sind unabhängig und*
- $H(X,Y) = H(Y) + H(X \mid Y)$, *wobei die Ausgabe der Quelle (X,Y) das Paar (x,y) mit der Wahrscheinlichkeit $p(x) \cdot p(y)$ ist.*

Beweis: S. z.B. Stinson, S. 56ff. ■

Angewandt auf ein Kryptosystem hat die bedingte Entropie einen speziellen Namen:

3.11. Definition *Sei $S = (\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem. Identifiziere $\mathcal{P}$, $\mathcal{C}$ und $\mathcal{K}$ mit Zufallsvariablen, die das Auftreten von Klartext, Chiffretext und Schlüssel beschreiben. Die bedingte Entropie $H(\mathcal{K} \mid \mathcal{C})$ wird* **Äquivokation** *von S genannt.*

$H(\mathcal{K} \mid \mathcal{C})$ mißt die Informationsmenge, die durch die Kenntnis des Chiffretextes über den Schlüssel gewonnen wird. Je kleiner $H(\mathcal{K} \mid \mathcal{C})$, desto „brechbarer" ist das Kryptosystem im Prinzip.

3.12. Satz *Es gilt*

$$H(\mathcal{K} \mid \mathcal{C}) = H(\mathcal{K}) + H(\mathcal{P}) - H(\mathcal{C}).$$

Beweis: Wir haben $H(\mathcal{K}, \mathcal{P}, \mathcal{C}) = H(\mathcal{K}, \mathcal{P}) + H(\mathcal{C} \mid (\mathcal{K}, \mathcal{C})) = H(\mathcal{K}, \mathcal{P}) = H(\mathcal{K}) + H(\mathcal{P})$. Analog gilt $H(\mathcal{K}, \mathcal{P}, \mathcal{C}) = H(\mathcal{K}, \mathcal{C})$. Also haben wir

$$\begin{aligned} H(\mathcal{K} \mid \mathcal{C}) &= H(\mathcal{K}, \mathcal{C}) - H(\mathcal{C}) \\ &= H(\mathcal{K}, \mathcal{P}, \mathcal{C}) - H(\mathcal{C}) \\ &= H(\mathcal{K}) + H(\mathcal{P}) - H(\mathcal{C}). \end{aligned}$$

■

Bezeichne N die Anzahl der empfangenen Schlüsseltextzeichen und $\mathcal{C}^N$ alle N-Gramme des Schlüsseltextes. Durch die statistischen Einflüsse des Klartextes sinkt $H(\mathcal{K} \mid \mathcal{C}^N)$ bei zunehmendem Schlüsseltextmaterial. Wir interessieren uns für das kleinste N, für welches $H(\mathcal{K} \mid \mathcal{C}^N)$ „nahe"Null ist, also das Kryptosystem (theoretisch) brechbar ist.

3.13. Definition *Sei $\mathbb{Z}_m$ ein endliches Alphabet und L eine Sprache über $\mathbb{Z}_m$. L_n bezeichne die Worte mit Länge n, es gilt $L = \bigcup_n L_n$.*

- *$R := \log|m|$ heißt* **absolute Dichte** *von L,*
- *$r_n := H(L_n)/n$ heißt* **n-te Dichte** *von L,*
- *$r = \lim_{n\to\infty} r_n$ heißt* **Dichte** *von L,*
- *$D = R - r$ heißt* **Redundanz** *von L und*
- *D/R heißt* **relative Redundanz** *von L.*

Die Dichte einer Sprache gibt die durchschnittliche Anzahl von Bits an, die man theoretisch benötigt, um einen Buchstaben zu kodieren. Im allgemeinen kann die Dichte einer (natürlichen) Sprache nicht berechnet werden; man nimmt statt dessen eine Schätzung für r_n für großes n. Diese stellt eine obere Schranke für r dar, da mit größer werdendem n die Entropie pro Buchstabe abnimmt.

3.14. Beispiel Sei L die englische Sprache ohne Sonderzeichen über $\mathbb{Z}_{26}$. Dann gilt für große n:

- $R = \log_2 26 \approx 4,7$
- $1,0 \leq r_n \leq 1.5$
- $3,2 \leq D \leq 3,7$
- $68\% \leq D/R \leq 79\%$

3.15. Definition *Sei $S = (\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem und D die Redundanz der Sprache der Klartexte. Dann heißt*

$$ED(S) := H(\mathcal{K})/D$$

Eindeutigkeitsdistanz *von S.*

3.16. Satz *Sei $S = (\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ ein Kryptosystem und D die Redundanz der Sprache der Klartexte. Nehmen wir an, daß alle Chiffretexte mit der gleichen Wahrscheinlichkeit auftreten. Ist der verwendete Schlüssel theoretisch (!) bestimmt, so werden mindestens $ED(S)$ viele Chiffretexte mit dem gleichen Schlüssel verschlüsselt und übermittelt.*

Beweis: Sei $c \in \mathcal{C}^N$. Bezeichne $K(c) := \{k \in \mathcal{K} : \exists x \in \mathcal{P}^N, p_{\mathcal{P}^N} > 0, e_k(x) = c\}$ und $s_N := \sum_{c\in\mathcal{C}^N} p(c) \cdot |K(c)|$ (durchschnittliche Anzahl der möglichen Schlüssel). Dann ist

$$\begin{aligned}
\log s_N = \log \sum_{c\in\mathcal{C}^N} p(c)|K(c)| &\geq \sum_{c\in\mathcal{C}^N} p(c)\log|K(c)| \text{da log konkav}\\
&\geq \sum_{c\in\mathcal{C}^N} p(c) H(\mathcal{K}_{|c}) = H(\mathcal{K} \mid \mathcal{C}^N).
\end{aligned}$$

Andererseits ist

$$\begin{aligned}
H(\mathcal{K} \mid \mathcal{C}^N) &= H(\mathcal{K}) + H(\mathcal{P}^N) - H(\mathcal{C}^N)\\
&\geq H(\mathcal{K}) + n \cdot r_n - n \cdot \log|\mathcal{C}|\\
&= H(\mathcal{K}) + n(r_n - R)\\
&\geq H(\mathcal{K}) + n(r - R)\\
&= H(\mathcal{K}) - n \cdot D.
\end{aligned}$$

Falls $s_N = 1$, ist $\log s_N = 0$, also $H(\mathcal{K} \mid \mathcal{C}^N) \leq 0$, bzw. $H(\mathcal{K}) - n \cdot D \leq 0$ also $n \geq H(\mathcal{K})/D$. ■

3.17. Bemerkung ED ist umgekehrt proportional zur Redundanz einer Sprache. Man kann den Einfluß der Redundanz einer Sprache reduzieren, indem man eine Huffman-Codierung vor die Verschlüsselung „vorschaltet".

3.18. Beispiel Annahme: $D = 3,2$, $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$ für die englische Sprache.

	Schlüsselanzahl	$H(\mathcal{K}) = \log(\lvert\mathcal{K}\rvert)$	ED
Affine Chiffre	312	$8,3$	$2,6$
Substitution	26!	$88,4$	$27,6$
Vigenère Chiffre (k)	26^k	$4,7 \cdot k$	$1,5 \cdot k$
DES	2^{56}	56	17,5

Tabelle 3.1: Beispiele für Eindeutigkeitsdistanzen

Kapitel 4

Data Encryption Standard - DES

1975 veröffentlicht von IBM

1977 als US-Standard für „unklassifizierte“ Anwendungen – wird ca. alle fünf Jahre neu beurteilt

1994 bis 1998 als Standard verabschiedet – wird voraussichtlich nicht verlängert

Es gilt: $\mathcal{P} = \mathbb{Z}_2^{64}, \mathcal{C} = \mathbb{Z}_2^{64}, \mathcal{K} = \mathbb{Z}_2^{56}$

4.1 Top-down-Beschreibung

4.1.1 High-level-Formulierung

Jeder String x ist geteilt in 2 32-Bit-Wörter $x = LR$

1. $L_0R_0 \longleftarrow IP(x)$ $\qquad$ $IP \in \sum_{64}$ fest gewählt (initial permutation)
2. $FOR\ i\ \leftarrow\ 1\ TO\ 16\ DO$
   $L_i \leftarrow R_i$
   $R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_i)$ $\qquad$ $\oplus : XOR, \quad f : \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{48} \longrightarrow \mathbb{Z}_2^{32}$
3. $y \leftarrow IP^{-1}(L_{16}R_{16})$

4.1.2 Beschreibung der Funktion f

$E : \mathbb{Z}_2^{32} \longrightarrow \mathbb{Z}_2^{48}$ fest

$S_i : \mathbb{Z}_2^{6} \longrightarrow \mathbb{Z}_2^{4}$ fest $\qquad i = 1, ..., 8$

$P : \mathbb{Z}_2^{32} \longrightarrow \mathbb{Z}_2^{32}$ fest

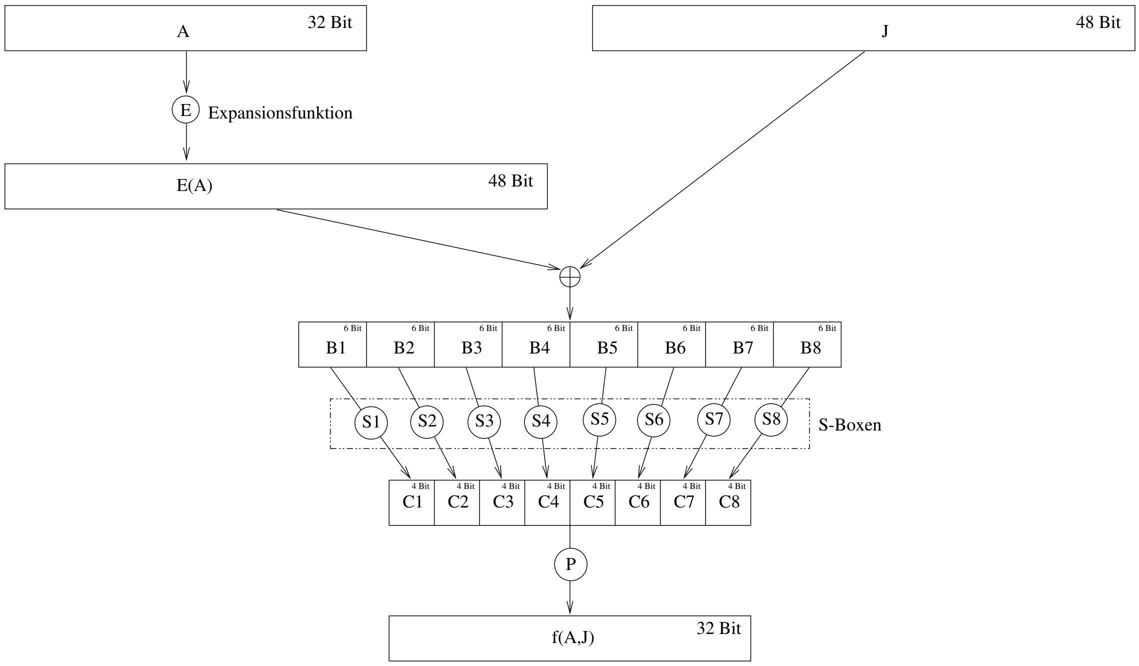


Abbildung 4.1: Schema des DES-Algorithmus

4.1.3 Beschreibung einer S-Box

$S_i \in \mathbb{Z}_{16}^{4\cdot 16}, S_i(B_j) = S_j(b_1 b_6, b_2 b_3 b_4 b_5)$

4.1.4 Berechnung aller 16 Rundenschlüssel

0. $K' \longleftarrow K$ entfernen der Paritätsbits
1. $C_0 D_0 \longleftarrow PC1(K')$
2. $FOR\ i\ \leftarrow 1\ TO\ 16\ DO$
   $C_i \leftarrow LS_i(L_{i-1})$
   $D_i \leftarrow LS_i(D_{i-1})$
   $K_i \leftarrow PC2(C_i D_i)$

LS_i: left-shift um $\begin{cases} 1Bit, falls\ i=1,2,9,16 \\ 2Bit\ sonst \end{cases}$
$PC2 \in \sum_{56} fest$

4.1.5 Entschlüsselung

wie Verschlüsselung mit Schlüsselreihenfolge $K_{16}, ..., K_1$.

4.2 Erste Analyse

Mit Ausnahme der S-Boxen sind alle Operationen **linear**, d.h. eine XOR-Berechnung von zwei Ausgaben ergibt dasselbe wie eine XOR-Berechnung der Eingaben und eine anschließende Berechnung der Ausgabe. Lineare Kryptosysteme sind (s. Kap. 2) leicht zu brechen.

Daher sind die S-Boxen, als nicht-lineare Komponenten von DES, entscheidend für die Sicherheit des Systems.

Auf die Vermutung, es würden NSA-Trapdoors in S-Boxen existieren, antwortete die NSA 1976 mit den **Design-Kriterien** für die S-Boxen.

- P0 Jede Zeile einer S-Box $\in \sum_{16}$.
- P1 Keine S-Box ist linear oder affin.
- P2 Die Änderung eines Input-Bits zieht die Änderung von mindestens zwei Output-Bits nach sich.
- P3 Für alle i und für alle x unterscheiden sich $S_i(x \oplus 001100)$ und $S_i(x)$ in mindestens zwei Bits.

Es ist nicht bekannt, ob dies alle Kriterien sind.

Ein weiterer Vorwurf besteht darin, der Schlüsselraum sei zu klein:

1977: Diffie & Hellman vermuten, daß es möglich ist, einen VLSI-Chip zu bauen, der 10^6 Schlüssel pro Sekunde testen kann. Der gesamte Schlüsselraum wäre in etwa einem Tag abgesucht. Geschätzte Kosten: 20 Millionen \$.

Crypto '93: Wiener stellt einen Chip vor, der 5×10^7 Schlüssel pro Sekunde testen kann und 10 \$ kostet. Eine Frame von 5760 Chips würde 100 000\$ kosten und den Schlüsselraum in 36 Stunden durchsuchen.

DES ist sehr effizient zu implementieren als Gatter oder Table-lookups. Sehr hohe Durchsatzraten ('92: 1GBit/sec (!) - im Vergleich RSA-Chip: 0,3 s für 512-Bit Exponentiation). Das Haupteinsatzgebiet sind Banken, PIN, ATM (Bankautomaten).

4.3 Genauere Analyse

4.1. Satz *DES ist invariant unter binärer Komplementbildung, d.h.* $Y = DES(X,K) \Rightarrow \overline{Y} = DES(\overline{X},\overline{K})$.

Beweis: Der Satz ist eine Konsequenz der folgenden Eigenschaften:

- $IP(\overline{X}) = \overline{IP(X)}$
- $E(\overline{R}) = \overline{E(R)}$ und $\overline{A} \oplus \overline{B} = A \oplus B$, also liefern $E(\overline{R}) \oplus \overline{K_i}$ und $E(R) \oplus K_i$ dieselbe Eingabe für die S_i, also $f(\overline{R},\overline{K_i}) = f(R,K_i)$
- $\overline{L_i} = \overline{R_{i-1}}$ und

- $\overline{R_i} = \overline{f(R_i, K_i)} \oplus \overline{L_{i-1}} = f(\overline{R_{i-1}}, K_i) \oplus \overline{L_{i-1}}$

■

Für die Sicherheit bedeutet dies keine wesentliche Einschränkung.

Nicht alle der 2^{56} möglichen Schlüssel sind gleich sicher:

4.2. Lemma *Es gibt genau 4 Schlüssel K mit der Eigenschaft $K_i = K_j$ für $i, j \in \{1, \ldots, 16\}$.*

Beweis: s. Übung. ■

Ein solcher Schlüssel heißt **schwacher Schlüssel**. Für solche Schlüssel gilt DES(DES(X,K),K)=X. Desweiteren gibt es 12 **semi-schwache Schlüssel**, welche paarweise folgende Eigenschaft erfüllen: DES(DES(X,K),K')=X.

Bei der Erzeugung von DES-Schlüsseln durch einen Zufallsprozeß ist die Wahrscheinlichkeit, einen solchen Schlüssel zu erhalten, sehr klein.

Als 64-Bit Blockfolge ist DES(X,K) $\in \sum_{64}$; somit sind DES-Iterationen definiert:

$$DES^n(X,K) = DES(DES^{n-1}(X,K),K) \qquad n \geq 2.$$

und es gibt $m \in \mathbb{N}$ mit

$$DES^m(X,K) = Id(X) = X.$$

4.3. Definition *$min\{m \in \mathbb{N} \mid DES^m(X,K) = X\}$ heißt* **Zykellänge** *oder* **Ordnung** *von DES(X,K).*

Die maximal mögliche Zykellänge ist 2^{64} und die minimale 2 (für die schwachen Schlüssel). Falls die Zykellänge für gegebene X und K klein ist, kann durch Iteration der Klartext leicht wiedergewonnen werden, die mittlere Zykellänge sollte also möglichst groß sein. Die wirkliche mittlere Zykellänge ist nicht bekannt, sie läßt sich nur unter der Annahme berechnen, daß sich DES wie eine zufällig gewählte Permutation auf $\mathbb{Z}_2^{64}$ verhält.

Bei gegebenem Schlüssel K und zufällig gewähltem Klartext X verhält sich also $DES(X,K)$ wie eine Permutation, deren Zykellänge jeden der Werte $2, \ldots, 2^{64}$ mit gleicher Wahrscheinlichkeit annimmt. Die Wahrscheinlichkeit, daß $DES(X,K)$ eine Zykellänge kleiner gleich einer vorgegebenen Schranke $r \in \{1, \ldots, 2^{64}\}$ besitzt, ist unter dieser Annahme damit $r \cdot \frac{1}{2^{64}}$, und die mittlere Zykellänge des DES ergibt sich zu $\frac{1}{2^{64}} \cdot \sum_{i=1}^{2^{64}} i = 2^{63} + \frac{1}{2}$. Obwohl kurze Zykel möglich sind, sind sie unwahrscheinlich, denn es treten z. Bsp. Zykel der Länge $10^6 (\approx 2^{20})$ oder kürzer mit der Wahrscheinlichkeit $\frac{1}{2^{44}}$ auf.

4.3.1 Der Avalanche-Effekt

„Kleine“ Änderungen im Klartext oder im Schlüssel führen zu „großen“Änderungen im Schlüsseltext. Die Abbildungen E und P sind mit dem Ziel konstruiert worden, Änderungen von wenigen Input-Bits in möglichst wenigen Runden über alle S-Boxen zu verteilen.

4.4 DES Betriebsarten

ECB Modus: (Electronic Code Book) Übliche Verwendung von DES als Blockchiffre, d.h. mehrere Klartexte werden mit demselben Schlüssel verschlüsselt.

CBC-Mode: (Cipher Block Chaining) Jeder Schlüsseltext y_i wird mit dem nächsten Klartext verXORd, bevor er mit demselben Schlüssel verschlüsselt wird. Also:
$y_0 \ = \ $ beliebiger Initialvektor,
$y_i \ = \ DES_k(y_{i-1} \oplus x_i), i \geq 1.$

OFB Modus: (Output Feedback) Ein Schlüssel wird wie folgt erzeugt:
$z_0 \ = \ $ beliebiger Initialvektor, $z_i \ = \ DES_k(z_{i-1}), \ i \geq 1.$
Die Verschlüsselung besteht dann aus:
$y_i \ = \ x_i \oplus z_i, i \geq 1.$

CFB Modus: (Cipher Feedback) Wie OFB-Modus, außer $z_i = DES_k(y_{i-1}), i \geq 1,$
$y_0 \ = \ $ beliebiger Initialvektor.

Variationen des CFB- und OFB-Modus: **k-feedback**-Modus. Dabei werden k Bits (oft k=1 oder 8) des Klartextes mit k Bits des Schlüsselstroms ver-xor-d; die Eingabe des DES wird um k Bits nach links geshiftet und mit den gerade verwendeten Schlüsselbits (OFB) bzw. dem gerade erzeugten Schlüsseltext (CFB) von rechts ergänzt.

ECB & OFB: Wird ein Klartext x_i geändert, wird nur der entsprechende Schlüsseltext davon beeinflußt. OFB wird oft zur Verschlüsselung von Satellitenübertragungen verwendet.

CBC & CFB: Wird ein Klartext x_i geändert, so sind alle nachfolgenden Schlüsseltexte davon betroffen. Diese Modi finden oft Verwendung bei **Authentifikation**.

MAC: (Message authentification code) An eine Folge von Klartexten wird eine Prüfsequenz angehängt, um den Empfänger (Bernd) von der Richtigkeit des Klartexte zu überzeugen (Authentizität). Diese Prüfsequenz wird **MAC** genannt.

Konstruktion eines MAC mit CBC-Modus:

- $x_1, \ldots, x_n \rightarrow DES_k \rightarrow y_1, \ldots, y_n$
- $MAC(x_1, \ldots, x_n) := y_n$
- Übermittle $x_1, \ldots, x_n, y_n$

Kennt Oskar den Schlüssel nicht, so ist es sehr unwahrscheinlich, daß er ein x_i und y_n so ändern kann, daß Bernd dies nicht bei der Entschlüsselung bemerkt.

4.5 Angriffe

Alle vorgestellten Attacken sind **chosen-plaintext**-Angriffe.

4.5.1 Time-Memory-Tradeoff

1. Gegeben ein Klartext-Schlüsseltextpaar: Probiere alle 2^{56} möglichen Schlüssel aus. Im Mittel werden 2^{55} Schüssel ausprobiert (außer: s.u.).
2. Gegen einen festen Klartext x, berechne alle möglichen $y = DES_k(x), k \in \mathbb{Z}_2^{56}$. Speichere diese Paare (y_k, k) in einer Tabelle, sortiert nach y_k. Bei Auftreten von $y = DES_k(x)$ suche y in der Tabelle. **Aufwand**: Konstante (Hashtabelle) bzw. logarithmische Zeit zum Suchen, aber max. Zeit und Platz (!) für die Vorberechnung.
3. Kombination beider Methoden (Hellmann 1980):
   Sei $R : \mathbb{Z}_2^{64} \longrightarrow \mathbb{Z}_2^{56}$ **Reduktionsfunktion** (beliebig)
   Bezeichne $g(k) = R(e_k(x)), \quad g : \mathbb{Z}_2^{56} \longrightarrow \mathbb{Z}_2^{56}$ für feste x.

Vorberechnung:

- Wähle R und m zufällige Schlüssel $K(i,0), 1 \leq i \leq m$
- Berechne $K(i,j) = g(K(i,l-1)), 1 \leq l \leq t$
- Speichere Paare $(K(i,t), K(i,0)), 1 \leq i \leq m$, sortiert nach der ersten Komponente.

Bezeichne $y_l = \begin{cases} R(y) & \text{falls } l = 1 \\ g(y_{l-1}) & \text{sonst.} \end{cases}$

4.4. Lemma $K = K(i, t-l) \Rightarrow y_l = K(i,t)$.

Beweis: $y_l = g^{l-1}(R(y)) = g^{l-1}(R(e_k(x)) = g^{l-1}(g(k)) = g^l(k)$.
$K(i,t) = g^l(K(i,t-l))$. ■

$\Leftarrow$ gilt nicht, da R, also g, nicht injektiv.

Algorithmus:
Eingabe: x, y, R (und DES-Verfahren)
Ausgabe: K mit $y = DES_k(x)$, falls $\exists i, j$ mit $K = K(i, t-j)$.

1. $y_1 \longleftarrow R(y)$

2. $FOR\ l \leftarrow 1\ TO\ t\ DO$

2.1. $IF\ y_l \in \{K(i,t) \mid 1 \leq i \leq m\}\ THEN$

2.1.1. $K \leftarrow g^{t-l}(K(i,0))$

2.1.2. $IF\ y = e_k(x)\ THEN\ RETURN\ (K)$

2.2. $y_{l+1} \leftarrow g(y_l)$

Komplexität:

1. Vorberechnung: Speicher $\mathcal{O}(m)$ Zeit $\mathcal{O}(mt)$
2. Algorithmus: Speicher $\mathcal{O}(m)$ Zeit $\mathcal{O}(t)$
   falls alle $K(i,j)$ verschieden
   (sonst maximal $\mathcal{O}(t^2)$)

und $P(K \in \{K(i,j)\}) = \frac{mt}{n}$, $n = 2^{56}$

Hellmann: Für $mt^2 = n$ gilt $P \geq 0,8 \cdot \frac{1}{t}$

Also: Wähle t „gute“ Reduktionsfunktionen und berechne t Tabellen vor.
Speicherbedarf: $\mathcal{O}(mt)$
Zeit: Vorbereitung: $\mathcal{O}(mt^2) = \mathcal{O}(n)$; Knacken: $\mathcal{O}(t^2)$
Time-Memory-Tradeoff (T-M-T): $m \approx t \approx n^{\frac{1}{3}} \approx 2^{19}$

Bemerkung: Time-Memory-Tradeoff hängt nicht von DES ab, möglich für alle Known-Plaintext-Angriffe.

4.5.2 Differentielle Kryptoanalyse

Ignoriere die (öffentlich bekannten) IP und IP^{-1}

Idee: Vergleiche das XOR von zwei Klartexten mit dem XOR der zugehörigen Schlüsseltexte.

4.5.3 S-Boxen

Sei S eine S-Box ($S = S_j$ für $1 \leq j \leq 8$). Für $B, B^* \in \mathbb{Z}_2^6$ bezeichne

- $B \oplus B^*$ den **Input-XOR** und
- $S(B) \oplus S(B^*) \in \mathbb{Z}_2^4$ den **Output − XOR**

von S. Für B' $\in \mathbb{Z}_2^6$ bezeichne $\Delta(B') := \{(B, B^*) : B' = B \oplus B^*\}$.

4.5. Lemma *Es gilt:*

1. $\Delta(B') = \{(B, B \oplus B'), B \in \mathbb{Z}_2^6\}$ *und*
2. $|\Delta(B')| = 2^6 = 64$.

Beweis:

1. „$\supseteq$“klar. „$\subseteq$“:
   $B \oplus B^* = B' \Rightarrow B^* = B' \oplus B^{-1}$ in der Gruppe $(Z_2^6, \oplus)$,
   $= B' \oplus B$ da B $\oplus$ B = 0.

2. wg. 1

■

Sei $B' \in \mathbb{Z}_2^6$ fest. Dann bezeichne

$$\begin{array}{rrcl} \varphi_{S,B'}: & \mathbb{Z}_2^6 & \longrightarrow & \mathbb{Z}_2^4 \\ & B & \longmapsto & S(B) \oplus S(B \oplus B') \end{array}$$

64 Input-XORs werden auf 16 mögliche Output-XORs abgebildet. Die Verteilung der Werte dieser Funktionen wird die Basis für den Angriff sein.

4.6. Beispiel $S = S_1, B' = 110100$

$c' \in \mathbb{Z}_2^4$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\mid \{\varphi_{S_1,B'}(B) = c' \mid B \in Z_2^6\} \mid$	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6

Für zufällig gewählte S und B' nimmt $\varphi_{S,B'}$ im Schnitt 75-80% der möglichen Werte an.

Es gibt 8 S-Boxen und 64 mögliche Werte für B', somit gibt es 512 Abbildungen $\varphi_{S,B'}$, also 512 Verteilungen, die man schnell (per Computer) berechnen kann.

Bezeichne $\varphi_{S,B'}^{-1}(C') = \{B \in \mathbb{Z}_2^6 : \varphi_{S,B'}(B) = C'\}$

(B ist jetzt $B \in \mathbb{Z}_2^6$. Es gilt nun wieder die Notation vom Anfang des Kapitels mit $B = B_1 \dots B_8, B_i \in \mathbb{Z}_2^6$ als Eingabe von S_i usw.)

$$\begin{array}{llll} B = E \oplus J, & E = E(R_{i-1}) & & \text{Expansion von } R_{i-1} \\ & J = K_i & & \text{Schlüsselbits für Runde i} \end{array}$$

$E_j, E_j^* \in \mathbb{Z}_2^6, C_j' \in \mathbb{Z}_2^4$. Bezeichne

$$T_j(E_j, E_j^*, C_j') := \{X \oplus E_j \mid X \in \varphi_{S_j, E_j \oplus E_j^*}^{-1}(C_j')\}$$

Diese Menge beinhaltet die potentiellen Schlüsselbits, die in der i-ten Runde verwandt wurden. Je mehr Tripel (E_j, E_j^*, C_j') zur Berechnung von T_j zur Verfügung stehen, desto weniger Möglichkeiten gibt es für die Schlüsselbits. Es gilt nämlich:

4.7. Satz *Sei $C_j' = S_j(B_j) \oplus S_j(B_j^*)$. Dann gilt: $J_j \in T_j(E_j, E_j^*, C_j')$.*

Beweis: $B \oplus B^* = (E \oplus J) \oplus (E^* \oplus J) = E \oplus E^*$, d.h. der Input-XOR hängt nicht vom Schlüssel ab.

$$\begin{aligned} B_j = E_j \oplus J_j, \quad & \text{also } J_j = B_j \oplus E_j \text{ (s.o.)} \\ & \text{und } \varphi_{S_j, E_j \oplus E_j^*}(B_j) \\ & = \varphi_{S_j, B_j \oplus B_j^*}(B_j) && = S_j(B_j) \oplus S_j(B_j \oplus (B_j \oplus B_j^*)) \\ & && = S_j(B_j) \oplus S_j(B_j^*) \\ & && = C'_j \end{aligned}$$

also $B_j \in \varphi^{-1}_{S_j, E_j \oplus E_j^*}(C'_j)$, also $J_j \in T_j(E_j, E_j^*, C'_j)$. ■

Bestimmt man N Tripel (E_j, E_j^*, C'_j), dann muß J_j im Schnitt der N Mengen $T_j(E_j, E_j^*, C'_j)$ liegen. Berechnung z.Bsp. durch Reference-Counter: 64 Zähler für die 64 möglichen Schlüssel. Inkrementiere für jedes gefundene Tripel (E_j, E_j^*, C'_j) die Zähler, die zu den Elementen in $T_j(E_j, E_j^*, C'_j)$ gehören. Hat man nach N Versuchen einen einzigen Zähler mit Wert N, so hat man den J_j-Teil des Schlüssels der i-ten Runde gefunden.

Angriff auf einen 3-Runden DES

Normal besitzt DES 16 Runden. Hier sollen zur Demonstration jedoch nur 3 verwendet werden. Wir ignorieren wieder IP und IP^{-1}.

Betrachte Paare von Klartexten und zugehörigen Schlüsseltexten: $L_0R_0, L_0^*R_0^*, L_3R_3, L_3^*R_3^*$. Es gilt:

$$\begin{aligned} R_3 &= L_2 \oplus f(R_2, K_3) = R_1 \oplus f(R_2, K_3) \\ &= L_0 \oplus f(R_0, K_1) \oplus f(R_2, K_3) \end{aligned}$$

Analog für R_3^*, also gilt:

$$R'_3 = R_3 \oplus R_3^* = L'_0 \oplus f(R_0, K_1) \oplus f(R_0^*, K_1) \oplus f(R_2, K_3) \oplus f(R_2^*, K_3)$$

Weiterhin $R_0 = R_0^*$, also $R'_0 = 0$
$\Rightarrow f(R_0, K_1) = f(R_0^*, K_1)$
$\Rightarrow f(R_2, K_3) \oplus f(R_2^*, K_3) = R'_3 \oplus L'_0$

Bezeichne C (resp. C^*) die Ausgabe der S-Boxen in der 3. Runde, also $f(R_2, K_3) = P(C)$ und $f(R_2^*, K_3) = P(C^*)$.

Also C' $= C \oplus C^* = P^{-1}(f(R_2, K_3)) \oplus P^{-1}(f(R_2^*, K_3))$
$= P^{-1}(f(R_2, K_3) \oplus f(R_2^*, K_3)) = P^{-1}(R'_3 \oplus L'_0)$ welches bekannt ist.

Andererseits gilt $L_3 = R_3$ (resp.$L_3^* = R_3^*$), also berechne $E = E(R_2) = E(L_2)$ und $E^* = E(R_2^*) = E(L_3^*)$.

Also: Jedes Quadrupel $(L_0R_0, L_0^*R_0^*, L_3R_3, L_3^*R_3^*)$ mit $R_0 = R_0^*$ führt zu einem Tripel (E, E^*, C'). Auf diese Weise Bestimmung der 48 Schlüsselbits für die 3. Runde im Schlüssel. Bestimmung der restlichen Bits durch Ausprobieren ($2^8 = 256$ Möglichkeiten).

Ein Angriff auf einen 6-Runden DES

Für den 3-Runden DES konnten wir einen beweisbaren Zusammenhang zwischen den L_i, R_i herstellen und die Bestimmung des Schlüssels auf die „Analyse“ einer S-Box reduzieren. Im allgemeinen geht das nicht.

4.8. Definition *Sei $n \in \mathbb{N}$. Eine Liste der Form*

$$L_0', R_0', R_1', p_1, \ldots, R_n', p_n$$

heißt **n-Runden Charakteristik**.
Seien $1 \leq i \leq n$ und R_{i-1} und R_{i-1}^* so, daß R_i, R_i^* mittels einer DES-Runde aus L_{i-1} hervorgehen. Dann ist die Wahrscheinlichkeit, daß $R_i \oplus R_i^* = R_i'$ gerade p_i.

Bemerkungen:

1. Man kann, der Vollständigkeit halber, L_i mit in die Liste aufnehmen. Da jedoch $L_i = R_{i-1}$ ist dies redundant.
2. Die Wahrscheinlichkeit, daß *gleichzeitig* alle $R_i \oplus R_i^* = R_i'$ ist *nicht* $p_1 \times \cdots \times p_n$, da die Schlüssel für die Runden nicht unabhängig voneinander sind. Trotzdem nehmen wir an, daß $p_1 \times \cdots \times p_n$ eine gute Näherung an diese Wahrscheinlichkeit ist.
3. Beachte, daß eine Charakteristik die Wahrscheinlichkeiten des Auftretens einer Folge für einen Klartext mit beliebigen Schlüsseln ausdrückt. Oskar hingegen wird zufällige Klartexte auswählen, um einen festen Schlüssel zu finden. Dabei nimmt er (wir) an, daß die aufgetretenen Wahrscheinlichkeiten nahe an den Wahrscheinlichkeiten der Charakteristik liegen.

4.9. Definition $p = p_1 \times \cdots \times p_n$ *heißt* **Wahrscheinlichkeit der Charakteristik**.

4.10. Beispiel (Alle Darstellungen in Hex-Code)

1. $L_0' = \text{egal}$
   $R_0' = 0_{16}$
   $R_1' = L_0'$
   $p = 1$

   <u>warum:</u>
   $R_1 = L_0 \oplus f(R_0, K_1)$
   $\Rightarrow R_1' = R_1 \oplus R_1^*$
   $\quad = L_0 \oplus f(R_0, K_1) \oplus L_1^* \oplus f(R_0^*, K_1)$
   $R_0' = 0, R_0 = R_0^*$
   $\Rightarrow f(R_0, K_1) = f(R_0^*, K_1)$
   $\Rightarrow R_1^* = L_0'$

   $\hookrightarrow$ Grundlage der o.g. 3-Runden-Attacke.

2. $L_0' = 00000000_{16}$, $R_0' = 60000000_{16}$, $R_1' = 00808200_{16}$, $p = \frac{14}{64}$.
   $E(R_0) \oplus E(R_0^*) = E(R_0 \oplus R_0^*) = C0000000_{16}$.
   Input-XOR von $S_1 = C$ $(S_2, \ldots, S_8 = 0)$, da Input-XOR von J nicht betroffen.
   Berechne $| \varphi_{S_1, C_k}(E_{16}) | = 14$. Somit ist der Output-XOR von S_1 1110 mit Wahrscheinlichkeit $\frac{14}{64}$. Also $C' = E0000000_{16}$.
   $P(C') = 00808200_{16}$ mit Wahrscheinlichkeit $\frac{14}{64}$. Somit:
   $R_1' = L_0' \oplus P(C') = 00808200_{16}$ mit Wahrscheinlichkeit $\frac{14}{64}$.

Der Angriff auf den 6-Runden DES basiert auf der folgenden 3-Runden Charakteristik:

$$
\begin{array}{ll}
L'_0 = 40080000_{16} & \\
R'_0 = 04000000_{16} & \\
R'_1 = 00000000_{16} & p_1 = \frac{1}{4} \\
R'_2 = 04000000_{16} & p_2 = 1 \\
R'_3 = 40080000_{16} & p_3 = \frac{1}{4}
\end{array}
$$

Analog zum 3-Runden Angriff gilt:
$R'_6 = L'_3 \oplus f(R_3, K_4) \oplus f(R_3^*, K_4) \oplus f(R_5, R_6) \oplus f(R_5^*, K_6)$

Mit Wahrscheinlichkeit $\frac{1}{16}$ (s. obige 2. Bemerkung) ist $L'_3 = R'_2 = 04000000_{16}$ und $R'_3 = 40080000_{16}$, also $E(R'_3) = 001000 \mid 000000 \mid 000001 \mid 010000 \mid 000000 \mid \ldots \mid 000000$. Somit sind die Input-XORs für $S_2,\ S_5,\ S_6,\ S_7,\ S_8$ (also auch für dessen Output-XORs) in Runde 4.

Bezeichne $C'_1 \ldots C'_8 = P^{-1}\ (R'_6 \oplus 04000000_{16})$. Mit Wahrscheinlichkeit $\frac{1}{16}$ werden $C'_2,\ C'_5,\ C'_6,\ C'_7,\ C'_8$ die Output-XORs von $S_2,\ S_5,\ S_6,\ S_7,\ S_8$ in Runde 6 sein (wegen obiger Gleichung).

Die Input-XORs dieser Runde können, analog zur 3-Runden-Attacke, aus den Chiffretexten bestimmt werden: $E_1 \ldots E_8 = E(R_5) = E(L_6)$.

Die Verteilung, die man durch Tripel (E_i, E_i^*, C'_i) für i=2,5,6,7,8 erhält, liefern aber zu $\frac{15}{16}$ zufälligen Müll statt mögliche Schlüssel. Wie kann man diesen Müll erkennen?

4.11. Definition *Sei $L_0 \oplus L_0^* = L'_0,\ R_0 \oplus R_0^* = R_0^*$ und $L'_0 R'_0 R'_1 p_1 \ldots R'_n p_n$ eine Charakteristik. Falls für alle i gilt: $R_i \oplus R_i^* = R_i$, dann heißt $L_0 R_0$* **richtiges Paar**, *sonst* **falsches Paar** *für die Charakteristik.*

Wie erwarten, daß ungefähr $\frac{1}{16}$ aller möglichen Klartexte ein richtiges Paar für unsere 3-Runden-Charakteristik bilden (s. obige 3. Bemerkung).

Falls wir mit einem richtigen Paar starten, dann gilt: $J_j = T_j(E_i, E_i^*, C'_i)$ für j=2,5,6,7,8.

Filterung: Falls $T_j(E_i, E_i^*, C'_i) = 0$ für ein j=2,5,6,7,8, so sind wir von einem falschen Paar ausgegangen. Ungefähr $\frac{2}{3}$ aller falschen Paare werden durch Filterung eliminiert. Nach dem Filterungsprozeß sind $\frac{1}{16}/\frac{1}{3} = \frac{3}{16}$ der Informationen brauchbar.

Die Bestimmung der korrekten Schlüsselbits erfolgt analog zum 3-Runden-DES-Angriff:
Es gibt $\prod_{j\in\{2,5,6,7,8\}} \mid T_j(E_i, E_i^*, C'_i) \mid$ mögliche **vorgeschlagene** Schlüsselbits pro Paar. Haben wir N Paare mit allen vorgeschlagenen Schlüsselbitstrings berechnet, so tritt der korrekte String ungefähr $\frac{3}{16}N$ mal auf, während falsche Strings eher zufällig auftreten, d.h. mit ungefähr Wahrscheinlichkeit $< \frac{1}{2^{30}} \ll \frac{3}{16}N$.

Die naive Berechnung aller Kombinationen von Schlüsselbitstrings ist zu kompliziert; wir verwenden wieder Reference-Counter T_j der Länge 64 zur Darstellung der $T_j(E_i, E_i^*, C'_i)$, wobei das i-te Bit 1 ist, falls $i \in T_j(E_i, E_i^*, C'_i)$, 0 sonst. Bezeichne T_j^l den Reference-Counter zu $T_j(E_i, E_i^*, C'_i)$ für das l-te Paar, $1 \leq l \leq N$.

$I \subseteq \{1, \ldots, N\}$ heißt **erlaubte Menge**, falls mindestens eine Koordinate von: $\sum_{i\in J} T_j^i$ gleich $\mid I \mid$ ist für alle $1 \leq j \leq 5$. Gibt es nur noch eine erlaubte Menge für ein $\mid I \mid$, sind wir fertig.

4.12. Beispiel 120 Paare, zufällig ausgewählt mit angegebenem XOR $\longrightarrow$ 120 Paare von Schlüsseltexten (hier ohne Filterung).

$\mid I \mid$	$\#I$
2	111
3	180
4	231
5	255
6	210
7	120
8	45
9	10
10	1 ☺

Filterung:
73 Paare werden aussortiert!

$\longrightarrow$ Bestimmung von 30 Bits des Schlüssels. Durch Anwendung dieses Verfahrens auf eine andere 3-Runden-Charakteristik bestimmt man weitere Bits, bis die Anzahl der fehlenden Schlüsselbits ≤ 15. Dann Ausprobieren.

Differentielle Kryptoanalyse für größere Rundenanzahl :

# Runden	8	10	12	14	16
#notwendige Klartextpaare	2^{14}	2^{24}	2^{31}	2^{39}	2^{47}

Kapitel 5

Public-Key-Kryptographie

5.1 Einwegfunktionen

$f : A \to B$ wollen wir Einwegfunktion nennen, wenn $f(x)$ für alle $x \in A$ leicht zu berechnen ist, aber $y = f(x)$ für fast alle $y \in B$ schwierig, bzw. nicht zu lösen ist.

1950 wurde eine Berechnung, die 10^6 Operationen und 10^4 Speicherplätze benötigt, als **sehr schwierig** angesehen. Heute wird eine Berechnung, die 10^{30} Operationen benötigt, als schwierig angesehen.

Physikalische Grenzen:

- 10^{60} Bit Massenspeicher werden nie konstruiert werden können (> Masse der Erde)
- 10^{70} Operationen werden nie ausgeführt werden können (> Energie der Sonne)

Definition abhängig vom Komplexitätsmaß.

5.1. Definition *Seien* $f : A \longrightarrow B,\ V, E \in \mathbb{N},\ p \in \mathbb{R},\ 0 \leq p \leq 1,\ K$ *ein beliebiges Komplexitätsmaß.* f *heißt* **Einwegfunktion der Charakteristik** (V, E, p) *genau dann, wenn:*

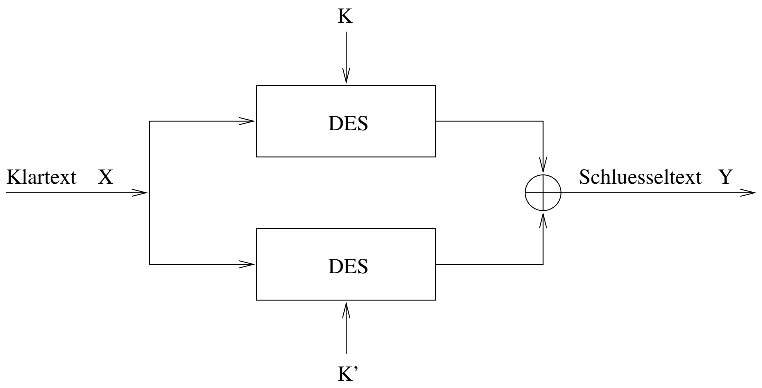
1. $K(f(x)) \leq V$, *für alle* $x \in A$
2. $P(K(Loesey) \leq E) \leq p$, *für gleichverteiltes* $x \in A$

K: (# Operationen) $\times$ (# Speicherplätze), dann ist eine Einwegfunktion der Charakteristik (10^6, $10^{130}, 10^{-70}$) praktisch unmöglich zu invertieren.

5.2. Beispiel

1. **Polynomfunktionen:**
   $g(x) := \sum_{i=0}^{n} a_i x^i,\ n \in \mathbb{N},\ a_i \in \mathbb{Z}_p$
   $f(x) := g(x)\ mod\ p$
   Berechne mit Horner-Schema in $\mathcal{O}$(log n) Multiplikationen. Noch besser: Dünn besetzt, d.h. $m = \#\{a_i \neq 0\} \ll n \longrightarrow \mathcal{O}$(log m). Für $p \geq 10^{20},\ n \geq 10^6$ Berechnung der Wurzel des Polynoms z.Zt. unmöglich.

2. **DES:**



3. **Multiplikation zweier Zahlen**

4. **Diskreter Logarithmus in einer endlichen abelschen Gruppe**

Für 1-4 ist nicht bekannt, ob es eine Charakteristik (V,E,p) gibt, mit 1-4 **beweisbar** Einwegfunktion der Charakteristik (V,E,p).

$\longrightarrow$ aktuelle Forschung der theoretischen Kryptographie.

5.2 Public-Key-Kryptosysteme

In den bisher vorgestellten (klassischen, symmetrischen) Kryptosystemen einigen sich Anke und Bernd auf einen gemeinsamen, geheimen Schlüssel. Dies hat zwei entscheidende Nachteile:

- d_k und e_k sind oft ähnlich, bzw. aus der Kenntnis von e_k läßt sich d_k gewinnen. D.h. e_k muß geheim bleiben.
- zwischen Anke und Bernd muß es einen geheimen, sicheren Kanal geben, über den der Schlüssel ausgetauscht wird.

Die Idee für ein Public-Key-Kryptosystem kam 1976 von Diffie & Hellmann und lautet wie folgt: Man stelle sich ein Kryptosystem vor, für welches es praktisch unmöglich ist, aus e_k d_k zu bestimmen. Dann kann Bernd e_k veröffentlichen; Anke kann ihm eine verschlüsselte Nachricht zuschicken und Bernd ist der einzige, der diese Nachricht entschlüsseln kann.

Public-Key Kryptosysteme (PKKS) können niemals perfekt sicher sein, da e_k bekannt ist und man somit alle Klartexte durchprobieren kann. Daher untersucht man die Sicherheit gegen Berechnung.

Abstrakt kann man ein PKKS als **Falltürfunktion** betrachten, d.h. ohne eine Zusatzinformation, die nur Bernd kennt, ist die Verschlüsselungsfunktion eine Einwegfunktion; Bernd kennt die Zusatzinformation, die Falltür, durch die er die Umkehrung schnell berechnen kann.

5.3 RSA

Wurde 1977 von Rivest, Shamir und Adleman entwickelt. Aus den ihren Initialen ergibt sich der Name für das Kryptosystem.

5.3. Definition *Sei* $n = p{\cdot}q$, *p und q Primzahlen. Sei* $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ *und* $\mathcal{K} = \{(n,p,q,a,b)\} \mid ab \equiv 1 \; mod \; (p-1)(q-1)\}$. *Für* $K = (n,p,q,a,b)$ *definiere:*

- $e_K(x) = x^b \; mod \; n$, *und*
- $d_K(x) = y^a \; mod \; n$.

$\mathcal{E} = \{e_k \mid K \in \mathcal{K}\}$, $\mathcal{D} = \{d_k \mid k \in \mathcal{K}\}$. *Dann heißt* $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ RSA-Kryptosystem.
n und b sind öffentlich bekannt.
p,q und a sind geheim.

Das Kryptosystem ist korrekt:

$$\begin{aligned} d_K(e_K(x)) &= d_K(x^b \; mod \; n) \\ &= x^{ab} \; mod \; n \\ &= x \; mod \; n \end{aligned}$$

(Weil: $ab = 1 \; mod \; (p-1)(q-1) = \phi(n)$, Kleiner Satz von Fermat $\Longrightarrow x^{ab} \equiv x^1 \; mod \; n$)

5.4. Beispiel $p = 101, q = 113 \rightarrow n = 11413, \phi(n) = (p-1)(q-1) = 11200, b = 3533$
Erweiterter Euklidischer Algorithmus $\longrightarrow a = b^{-1} = 6597 \; mod \; 11200$.
Veröffentliche n und b.

Anke will 9726 zu Bernd schicken, also schicke $9726^{3533} \mod 11413 = \underline{5761}$ über den öffentlichen Kanal.
Bernd berechnet den Klartext: $5761^{6597} \mod 11413 = 9726$. ☺

Wähle die beiden Primzahlen groß genug. Z.Bsp. gilt $\log_2 p = 256 \Rightarrow \log_{10} p = 77 \Rightarrow \log_{10} n = 154 \Rightarrow$ nicht lange sicher.

Für eine sichere Implementation sollte mindestens $log_2 p = log_2 q = 512$ gelten. Doch schon für $log_2 n = 512$ Bit ergibt sich ein Durchsatz von 600KB/s, also 1500 mal langsamer als DES. ☹

Die Konstruktion von RSA basiert auf Primzahltests sowie -beweisen. Seine Sicherheit erhält das Verfahren durch die Schwierigkeit der Faktorisierung großer Zahlen. Der folgende Algorithmus zeigt dies.

Algorithmus:
Eingabe: n, a, b $\quad ab \equiv 1 \ mod \ \phi(n)$
Ausgabe: $p \mid n$ mit $p \notin \{1, n\}$ oder „Pech!“

1. IF ggT$(w,n) \neq 1$ RETURN (w)
2. Berechne r, s mit $ab - 1 = 2^s \cdot r$ $\quad$ r ungerade
3. $v \leftarrow w^r \ mod \ n$
4. IF $v \equiv 1 \ mod \ n$ RETURN („Pech!“)
5. WHILE $v \not\equiv 1 \ mod \ n$ DO
6. $\quad v_0 \leftarrow v \ , \ v \leftarrow v^2 \ mod \ n$
7. IF $v_0 \equiv -1 \ mod \ n$ THEN RETURN („Pech!“)
8. RETURN (ggT$(v_0 + 1$,n))

Beweis: Korrektheit: Falls nicht „Pech!“ ausgegeben wird, so gilt $v_0 \not\equiv -1 \ mod \ n$ und $v_0 \equiv 1 \ mod \ n$, also $n = (v_0^2 - 1) = k \cdot (v_0 - 1)(v_0 + 1)$ und gleichzeitig $n \nmid (v_0 + 1), (v_0 - 1) \Rightarrow (v_0 + 1) \neq n$. Nach Konstruktion gilt $1 < v_0 + 1 < n$. Also $ggT(v_0 + 1, n)$ ist ein echter Teiler von n. ■

Eine genaue Analyse zeigt, daß der Algorithmus (Monte-Carlo-Typ) mit $W > \frac{1}{2}$ n faktorisiert.

Information über Klartextbits:

Bezeichne für $y = e_K(x)$ mit parity(y) das LSB (least significant bit) und mit half(y) das MSB (most significant bit) von x.

5.5. Lemma *parity(y) berechnen ist polynomial äquivalent zu half(y) berechnen.*

Beweis: $half(y) = parity(y \cdot e_K(2))$; $\quad parity(y) = half(y \cdot e_K(2^{-1}))$
Beweis von $\Rightarrow$ ($\Leftarrow$ äquivalent):
$half(y) = 1 \Leftrightarrow \frac{n}{2} \leq n - 1 \ (n \ ungerade)$
$parity(y \cdot e_K(2)) = parity(e_K(x) \cdot e_K(2)) = parity(e_K(2x)) = 1.$ ■

5.6. Satz *Gegeben ein Orakel, welches für $z \in \mathcal{C}$ $half(z)$ berechnet, kann man in polynomieller Zeit $d_K(y)$ berechnen.*

Beweis: Beachte:

$$half(e_K(x)) = 0 \Leftrightarrow x \in [0, \tfrac{n}{2})$$
$$half(e_K(2x)) = 0 \Leftrightarrow x \in [0, \tfrac{n}{4}) \cup [\tfrac{n}{4}, \tfrac{3n}{4})$$
$$half(e_K(4x)) = 0 \Leftrightarrow x \in [0, \tfrac{n}{8}) \cup [\tfrac{n}{4}, \tfrac{3n}{8}) \cup [\tfrac{n}{2}, \tfrac{5n}{8}) \cup [\tfrac{3n}{4}, \tfrac{7n}{8}),$$

usw. Also berechne $y_i = half(y \cdot (e_K(2))^i)$ für$i = 0, \ldots, \lfloor log_2 n \rfloor$ und bestimme dadurch x in $log_2 n$ Schritten. ■

5.4 El-Gamal Kryptosystem

5.7. Definition *Sei p eine Primzahl, $\alpha \in \mathbb{Z}_p^*$ ein primitives Element (d.h. ord(α) = p-1). Sei $\mathcal{P} = \mathbb{Z}_p^*, \mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ und*

$$\mathcal{K} = \{(p,\alpha,a,\beta) : \beta = \alpha^a (\bmod p)\}$$

Für $K = (p,\alpha,a,\beta) und k \in_R \mathbb{Z}_{p-1}$ seien ($x \in P$)

$$y_1 := \alpha^k \mod p$$
$$y_2 := x \cdot \beta^k \; mod \; p$$

und $e_K(x,k) = (y_1, y_2)$

sowie $\mathcal{E} = \{e_K \mid K \in \mathcal{K}\}, \mathcal{D} = \{e_K \mid K \in \mathcal{K}\}$.

Dann heißt ($\mathcal{P}$,$\mathcal{C}$,$\mathcal{K}$,$\mathcal{E}$,$\mathcal{D}$) El-Gamal-Kryptosystem.

Korrektheit:

$$\begin{aligned} d_K(e_K(x,k)) &= d_K(\alpha^k \; mod \; p, x\beta^k \; modp) \\ &= x\beta^k((\alpha^k)^a)^{-1} \; mod \; p \\ &= x\beta^k(\alpha^{-ak}) \; mod \; p \\ &= x((\tfrac{\beta}{\alpha^a})^k) \; mod \; p \\ &= x \qquad \beta = \alpha^a (mod \; p) \end{aligned}$$

$k \in Z_{p-1}$, damit folgende Idee nicht funktioniert: (chosen-plaintext)
$x = \alpha^l$ für mehrere l bis $x\beta \equiv 1, -1 \; mod \; n \longrightarrow$ Bestimmung von a!

5.5 Diskreter Logarithmus

5.8. Definition *Sei p eine Primzahl, $\alpha \in \mathbb{Z}_p^*$ primitives Element, $\beta = \alpha^a$. Dann heißt a Diskreter Logarithmus (DL) von β zur basis $\alpha : a = log_\alpha \beta$.*

Bislang ist noch keine polynomielle Berechnung des DL bei gut gewähltem p bekannt - vielmehr handelt es sich dabei um einen Gegenstand der aktuellen Forschung.

Bit-Sicherheit des Diskreten Logarithmus

5.9. Satz *p Primzahl, $\alpha \in \mathbb{Z}_2^*$ primitiv, $\beta = \alpha^n, n = log_\alpha \beta$. Das LSB eines Diskreten Logarithmus kann in $\mathcal{O}$(log p) Operationen in $\mathbb{Z}_p^*$ berechnet werden (d.h. es ist einfach zu bestimmen).*

Beweis: Behauptung: $\mid QR(p) \mid := \mid \{x^2 modp\} \mid = \frac{p-1}{2}$
$w^2 = x^2 \; mod \; p \Leftrightarrow p \mid (w-2)(w+x) \Leftrightarrow w \equiv \pm x (mod \; p)$
$\Rightarrow \forall y \in \mathbb{Z}_p^* \; \exists! x \in \mathbb{Z}_p^2 \; x^2 = y \; mod \; p \longrightarrow$ Behauptung.
Sei $\alpha \in \mathbb{N}$. a gerade $\Rightarrow \alpha^a \equiv (\alpha^{\frac{a}{2}})^2 \; mod \; p \Rightarrow \alpha^a \in QR(p)$.
Da $\alpha^{2i}, i = 0, \ldots, \frac{p-3}{2}$ alle verschiedenen mod p
$\Rightarrow$ a gerade $\Longrightarrow \alpha^a \in QR(p)$. Also $\beta \in QR(p) \Leftrightarrow LSB(n) = 0$. Nun gilt die Euler-Formel :

$$\beta^{\frac{p-1}{2}} \equiv 1 \; mod \; p \Leftrightarrow \beta \in QR(p)$$

Also:

$$LSB(n) = \begin{cases} 0, & \text{falls } \beta^{\frac{p-1}{2}} \equiv 1 \; mod \; p \\ 1 & sonst \end{cases}$$

$\beta^{\frac{p-1}{2}} \; mod \; p$ kann in $\mathcal{O}$(log p) Operationen in $\mathbb{Z}_p^*$ durch binäre („schnelle“) Exponentiation berechnet werden. ■

Man kann analog zeigen, daß das i-te LSB einfach berechnet werden kann, wenn $i \leq s$, wobei $p-1 = 2^s t, 2 \not| t$ (mit einer analogen Technik).

Das $s+1$-te LSB zu berechnen ist mindestens so schwierig, wie den DL insgesamt zu berechnen (s.u.).

5.10. Satz *Sei p eine Primzahl, $p \equiv 3 \mod 4$ und $p = 1 = 2t$, t ungerade. Sei $\alpha \in Z_p^*$ primitiv und $\beta = \alpha^n$. Dann kann man $n = log_\alpha \beta$ in polynomieller Zeit in log_p berechnen, wenn man ein Orakel für das 2-te LSB hat.*

Beweis: $i - LSB(\alpha) = i$-tes Bit von $x_n : x = \alpha^{xm}$

Beachte: 1) $x^2 \equiv \beta \; mod \; p \Rightarrow x = \pm \beta^{\frac{p+1}{4}} \; mod \; p$
und 2) $LSB(\beta) \neq LSB(p-\beta)$, denn
$-\beta \equiv \alpha^{a+\frac{p-1}{2}} \; mod \; p$ und $\frac{p-1}{2}$ ungerade.

1. Schritt: $LSB(\beta)$ berechnen (s.o.), jetzt a gerade

2. Schritt: bestimme 2-LSB(β) mit dem Orakel.
Nun ist $2 - LSB(\beta) = LSB(\alpha^{\frac{a}{2}})$, also kann man $\alpha^{\frac{a}{2}}$ berechnen:
$\alpha^{\frac{a}{2}} \equiv \pm \beta^{p+1} 4 \; mod \; p$ $\qquad \pm$durch$LSB(\alpha^{\frac{a}{2}}$ bestimmt.

3. Schritt: Setze $\beta \leftarrow$ „$\alpha^{\frac{a}{2}}$“und iteriere das Verfahren mit Schritt 2.

Die Anzahl der Iterationen beträgt $\mathcal{O}$(log p). ■

5.6 Verallgemeinerung des DL

Allgemeiner kann man den DL in einer beliebigen endlichen abelschen Gruppe definieren und somit nach Gruppen suchen, die für kryptographische Anwendungen besonders geeignet erscheinen.

- Endliche Körpererweiterungen $\mathbb{F}_{p^l}$ von $\mathbb{F}_p$, insbesonders für p = 2.
- Elliptische Kurven über $\mathbb{F}_p(\mathbb{F}_{p^l})$
- Klassengruppen von Zahlkörpern
- Divisorklassengruppen von hyperelliptischen Kurven
- „Infrastruktur“in reellquadratischen Zahlkörpern

5.7 Andere Kryptosysteme

Ansatz: Nehme ein NP-vollständiges Problem und benutze eine (einfache) Instanz mit Lösung als Public-key/Privat-key-Paar.

Beispiele:

1) Merkle-Hellmann-Knapsack basiert auf dem „Subset-Sum-Problem“: Bestimme $x_1, \ldots, x_n$ sodaß $\sum_{i=1}^{n} x_i s_i \;=\; T$.
Anfang der 80'er Jahre geknackt (von Shamir & Lenstra)

2) McEliece basiert auf dem Problem: „Dekodieren eines allg. linearen fehlerkorr. Codes“.

Kapitel 6

Signaturschemata

Signieren bedeutet im allgemeinen „Unterschreiben“. Hier bezeichnet es das Erzeugen einer „elektronischen Unterschrift“.

Probleme:

- Bei einer konventionellen Unterschrift eines Dokuments ist die Unterschrift physikalischer Teil des Dokuments. Das ist bei einer elektronischen Unterschrift a priori nicht so. Man muss diese an das elektronische Dokument „binden“.
- Wie bei einer konventionellen Unterschrift muß jeder die Unterschrift nachprüfen können.
- Bei einer konventionellen Unterschrift ist die „Sicherheit“ gegen Mißbrauch durch Dritte dadurch gekennzeichnet, daß man eine Unterschrift mit anderen, schon bekannten vergleicht und nachgemachte Unterschriften auffallen (sollen). Elektronische „Kopien“ sind identisch, daher muß das zu signierende Dokument in die Unterschrift mit eingehen, um Mißbrauch zu vermeiden.
- Das Signaturgesetz ist im Juli 1997 verabschiedet worden.

6.1. Definition *Ein* **Signaturschema** *ist ein 5-Tupel* $(\mathcal{P},\mathcal{A},\mathcal{K},\mathcal{S},\mathcal{V})$ *von endlichen Mengen mit:*

$$\begin{array}{ll} \forall K \in \mathcal{K}\ \exists sig_k \in \mathcal{S}, ver_k \in \mathcal{V} & sig : \mathcal{P} \to \mathcal{A} \\ \forall x \in P\ \forall y \in \mathcal{A} & ver : \mathcal{P} \times \mathcal{A} \to \{T, F\} \\ \quad ver(x,y) \ = \ T \Leftrightarrow y = sig(x) & \end{array}$$

$\mathcal{P}$ ist die Menge der Nachrichten (Dokumente)
$\mathcal{A}$ ist die Menge der Unterschriften (Signaturen)
$\mathcal{K}$ ist die Menge der Schlüssel
$\mathcal{S}$ ist die Menge der Signaturen
$\mathcal{V}$ ist die Menge der Überprüfungen (Verifikationen)

Bernd unterschreibt ein Dokument mit einer Signierung sig_k, Anke kann die Unterschrift mit der zugehörigen Verifikation ver_k überprüfen, ver_k ist öffentlich und sig_k ist geheim. Sowohl ver_k als auch sig_k sollten effizient berechenbar sein. Oskar sollte praktisch nicht in der Lage sein, Bernd's Unterschrift auf einem Dokument x nachzumachen, d.h. y so zu berechnen, daß $ver_k(x,y) \ = \ T$.

Offensichtlich kann kein Signaturschema perfekt sicher sein, denn mit genug Zeit kann Oskar alle y durchprobieren und testen ob $ver_k(x,y) = T$.

Hier fehlt noch das RSA und ElGamal-Signaturverfahren
(Mitschrift aus der Vorlesung bitte als Kopie zum Autor)

Das Verfahren ist korrekt:

1. Sei $(\gamma,\delta) = sig_k(x) = (\alpha^k(mod\ p),(x-a\gamma)k^{-1}(mod\ p-1))$
   Dann ist $\beta^\gamma\gamma^\delta \equiv (\alpha^a)^\gamma(\alpha^k)^\delta \equiv \alpha^{a\gamma+k\delta} \equiv \alpha^{a\gamma+k(\frac{x-a\gamma}{k})} \equiv \alpha^x(mod\ p)$

2. Sei $\beta^\gamma\gamma^\delta \equiv \alpha^x(mod\ p)$,also$(\alpha^a)^\gamma\gamma^\delta \equiv \alpha^x(mod\ p)$,d.h.$\gamma^\delta \equiv \alpha^{x-a\gamma}(mod\ p)$. Da α primitiv, gibt es genau ein $k \in \mathbb{Z}_{p-1}$ mit $\gamma = \alpha^k$, also $(\alpha^k)^\delta \equiv \alpha^{x-a\gamma}(mod\ p) \Leftrightarrow k\delta \equiv x - a\delta(mod\ p-1)$

6.2. Beispiel $p = 467, \alpha = 2, a = 127 \Rightarrow \beta = \alpha^a\ mod\ p = 2^{127}\ mod\ 467 = 132$
$x = 100, k = 213 \qquad (ggT(213,466) = 1, k^{-1} = 431)$
$\Rightarrow \gamma = 2^{213}\ mod\ 467 = 29$
$\delta = (100 - 127 \times 29)431\ mod\ 466 = 51$
Check: $132^{29}29^{51} \equiv 189(mod\ 467)$ und $2^{100} \equiv 189(mod\ 467)$

Sicherheit: Oskar will eine Nachricht als von Bernd unterschrieben erzeugen.

1. Ansatz: Wähle $\gamma \Longrightarrow$ berechne $log_\gamma \alpha^x\beta^{-\gamma}$.

2. Ansatz: Wähle $\delta \Longrightarrow$ löse $\beta^\gamma\gamma^\delta \equiv \alpha^x(mod\ p)$ nach γ.

3. Ansatz: Wähle (γ,δ) gleichzeitig:

Mit „zufälligem“Dokument: $0 \leq i \leq p-2, 0 \leq j \leq p-2, ggT(j,p-1) = 1$

$$\begin{aligned} \gamma &= \alpha^i\beta^j(mod\ p), \\ \delta &= -\gamma j^{-1}(mod\ p-1), \\ x &= \gamma i j^{-1}(mod\ p-1) \end{aligned}$$

$\Longrightarrow (x,(\gamma,\delta))$ ist eine gültige Signatur

Mit vorher bekanntem Sign.-Dokument: $(x,(\gamma,\delta))$
$0 \leq h,i,j \leq p-2, ggT(h\gamma - i\delta, p-1) = 1$

$$\begin{aligned} \lambda &= \gamma^h\alpha^i\beta^j(mod\ p) \\ \mu &= \delta\lambda(h\gamma - j\delta)^{-1}(mod\ p-1) \\ x' &= \lambda(hx + i\delta)(h\gamma - j\delta)^{-1}(mod\ p-1) \end{aligned}$$

$\Longrightarrow (x',(\lambda,\mu))$ ist eine gültige Signatur

Beide Angriffe mit „sinnloser“Nachricht $\Longrightarrow$ keine Beeintrchtigung der praktischen Sicherheit.

k muß geheim gehalten werden, denn sonst kann man $a = (x-k\gamma)\delta^{-1}\ mod\ (p-1)$ leicht berechnen und das Signatursystem ist geknackt.

Aufgabe: Angenommen Bernd verwendet zweimal hintereinander das gleich k und Oskar weiß das. Wie kann Oskar daraus k berechnen und damit (s.o.) das Signatursystem knacken?

6.1 Digital Signature Standard (DSS)

Wurde 1991 vorgeschlagen und 1994 modifiziert und als Standard angenommen.

DSS ist eine Modifikation des ElGamal Signatursystems.

6.3. Definition *Sei p eine Primzahl mit 512 Bits und q eine 160Bit Primzahl mit $q \mid p-1$. Sei $\alpha \in \mathbb{Z}_p^*$ eine q.te Einheitswurzel* $\bmod p$. *Sei* $\mathcal{P} = \mathbb{Z}_p^*, \mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$ *und*
$\mathcal{K} = \{(p,q,\alpha,a,\beta),\ \beta \equiv \alpha^a (\bmod\ p)\}$
Für $K = (p,q,\alpha,a,\beta)$ *und* $k \in_R \{1,\ldots,q-1\}$ *seien*

$$sig_K(x,k) = ((\alpha^k \bmod p) \bmod q, (x + a\gamma)k^{-1} \bmod q)$$

und $ver_K(x,(\gamma,\delta)) = T^\gamma$

$$\Leftrightarrow (\alpha^{e_1}\beta e_2 \bmod p) \bmod q = \gamma$$
$$mit e_1 = x\delta^{-1} \bmod q$$
$$e_2 = \gamma\delta^{-1} \bmod q.$$

Sei $\mathcal{S} = \{sig_K : K \in \mathcal{K}\}, \mathcal{V} = \{ver_K : K \in \mathcal{K}\}$
Dann ist $(\mathcal{P},\mathcal{A},\mathcal{K},\mathcal{S},\mathcal{V})$ *ein* **Signaturschema nach DSS**.

$\delta \equiv 0 \bmod q$ muß von Bernd vermieden werden, sonst kann die Signatur nicht überprüft werden $\Longrightarrow$ wähle dann ein neues k. Dies passiert jedoch selten: $\sim 2^{-160}$.

6.4. Beispiel $q = 101, p = 78 \cdot q + 1 = 7879$.
$\alpha = 3^{78} \bmod 7879 = 170. \qquad (<3> = \mathbb{Z}_{7879}^*)$
$a = 75 \Rightarrow \beta = \alpha^a \bmod 7879 = 4567.$

$x = 1234, k = 75 \Longrightarrow k^{-1} \bmod 101 = 99$

$$\begin{aligned}\gamma &= (170^{50} \bmod 7879) \bmod 101.\\ &= 2518 \bmod 101 = 94\\ \delta &= (1234 + 75 \cdot 94) \cdot 99 \bmod 101 = 97.\end{aligned}$$

<u>Verifikation:</u>
$\delta^{-1} = 97^{-1} \bmod 101 = 25, e_1 = 45, e_2 = 27$
$(170^{45} \cdot 4567^{27} \bmod 7879) \bmod 101 = 2518 \bmod 101 = 94.\surd$

Vorteil: 512-Bit-Nachricht wird mit 320Bit-Schlüssel versehen (vgl. ElGamal: 512 vs. 1024 !!)

Idee: Betrachte den DL in einer Untergruppe von $\mathbb{Z}_p^*$ der Größe $\approx 2^{160}$. Die Berechnungen finden nach wie vor in $\mathbb{Z}_p$ statt.

512Bit-Basisgroße wurde später durch $512 \leq l \cdot 64 \leq 1024, l \in \mathbb{N}$ ersetzt.

Kapitel 7

Hashfunktionen

Probleme bei Signaturschemata: Nur „kleine“ Nachrichten können signiert werden. Bsp.: ElGamal: Länge der Signatur = 2 × Länge der Nachricht!

Lösung → Hashfunktionen: Öffentlich bekannte Schrumpfabbildungen, die aus einer Nachricht beliebiger Länge eine Nachricht vorgegebener Länge (z.B. 512 Bit) machen.

Allgemeine Vorgehensweise:
Aus einer zu signierenden Nachricht x berechnet Bernd mit der öffentlichen Hashfunktion h $h(x)$, wofür er eine Signatur erstellt $y = sig(h(x))$. Er sendet dann das Paket $(x, sig(h(x)))$. Das Checking geschieht durch Anwendung von h auf x und anschließendes überprüfen, ob $ver(h(x), sig(h(x))) = T$.

7.1 Kollisionsfreiheit

Die Verwendung einer Hashfunktion h darf die Sicherheit eines Signaturschemas nicht beeinträchtigen. Der offensichtliche Angriff, der durch Verwendung einer Hashfunktion möglich ist, sieht wie folgt aus:

Oskar kennt eine signierte Nachricht $(x, y), y = sig(h(x))$. Er berechnet $z = h(x)$ und versucht ein sinnvolles x' zu finden, so daß $h(x') = z$. Dann ist (x', y) als eine (von Bernd) unterschriebene Nachricht zu erkennen.

Dies führt zu folgender

7.1. Definition *Eine Hashfunktion h ist* **schwach kollisionsfrei**, *wenn es für gegebenes x praktisch unmöglich ist, ein x' zu finden, mit $h(x) = h(x')$.*

Anderer Angriff: Oskar findet zwei Texte x und x' mit $h(x) = h(x')$ und überzeugt Bernd davon, $h(x)$ zu signieren. Dann ist wiederum (x', y) als eine vom Bernd signierte Nachricht zu erkennen.

7.2. Definition *Eine Hashfunktion h ist* **stark kollisionsfrei**, *falls es praktisch unmöglich ist, x und x' zu finden, mit $h(x) = h(x')$.*

Offensichtlich beinhaltet die starke Kollisionsfreiheit die schwache.

Weiterer Angriff: Bestimmt zufälliges z, lasse dieses signieren und finde dann x mit $h(x) = z$.

7.3. Definition *Eine Hashfunktion h ist* **nicht umkehrbar**, *wenn es zu gegebenem z praktisch unmöglich ist ein x zu finden, mit $h(x) = z$.*

7.4. Satz *Sei $h : X \to Z$ eine Hashfunktion mit $|X|, |Z| < \infty$ und $|X| \geq n|Z|$. Sei A ein Algorithmus, welcher zu gegebenem $z \in Z$ ein $x \in X$ berechnet, mit $h(x) = z$. Dann gibt es einen Las Vegas-Algorithmus, welcher Kollisionen findet mit Wahrscheinlichkeit $\geq \frac{1}{2} - \frac{1}{n}$.*

($\longrightarrow$ stark kollisionsfrei $\Rightarrow$ nicht umkehrbar)

Beweis: Betrachte Algorithmus B:

1. $x \in_R X, z \leftarrow h(x), x_1 = A(z)$

2. $IF\ x_1 \neq x\ THEN\ RETURN\ (x_1, x)$
$ELSE\ RETURN()$

Behauptung:$Wahrscheinlichkeit(Ausgabe = (x_1, x)) \geq 1 - \frac{1}{n}$
$h^{-1} := \{x_1 \in X \mid h(x_1) = z\}$
Für eine gegebenes x liefert $A(h(x))$ eine Kollision, falls $x \neq A(h(x))$, also mit Wahrscheinlichkeit $\frac{|h^{-1}(h(x))|-1}{|h^{-1}(h(x))|}$.
Also findet Algorithmus B eine Kollision mit Wahrscheinlichkeit

$$\begin{aligned}
\frac{1}{|X|}\sum_{x\in X}\frac{|h^{-1}(h(x))|-1}{|h^{-1}(h(x))|} &= \frac{1}{|X|}\sum_{z\in Z}\left(\underbrace{\sum_{x\in h^{-1}(z)}\frac{|h^{-1}(h(z))|-1}{|h^{-1}(h(z))|}}_{|h^{-1}(z)|-1}\right) \\
&= \frac{1}{|X|}\left(\sum_{z\in Z}|h^{-1}(z)| - -\sum_{z\in Z}1\right) \\
&= \frac{1}{|X|}(|X|-|Z|) \geq \frac{1}{|X|}\left(|X| - \frac{|X|}{n}\right) = 1 - \frac{1}{n}
\end{aligned}$$

■

7.2 Geburtstagsangriff (Birthday attack)

Birthday „paradoxon“: Unter 23 zufällig ausgewählten Menschen haben 2 am gleichen Tag Geburtstag, mit Wahrscheinlichkeit $\geq \frac{1}{2}$.

Sei $h : X \to Z, |X|, |Z| < \infty$. Ist $|X| \geq 2|Z|$, so gibt es mindestens $|Z|$ Kollisionen. Wie findet man diese?

Idee: Wähle k zufällige $x_i \in X$, berechne $h(x_i)$ und prüfe, ob Kollisionen vorliegen. Wie erfolgreich ist dieses Verfahren?

Annahme:$|h^{-1}(z)| \approx \frac{|X|}{|Z|}$. Ist dies nicht der Fall, dann ist die Wahrscheinlichkeit, eine Kollision zu finden, noch größer.

7.5. Satz *Voraussetzung wie oben. Dann ist die Wahrscheinlichkeit, eine Kollision durch Wahl von k zufälligen Klartexten zu bestimmen:*

$$\approx 1 - e^{\frac{-k(k-1)}{|Z|}}.$$

Wählt man $\geq \sqrt{|Z| ln \frac{1}{1-\epsilon}}$ *zufällige Klartexte, so ist die Wahrscheinlichkeit einer Kollision* $\geq \epsilon$.

7.6. Beispiel $\epsilon = 0,5, |Z| = 365 \longrightarrow k \geq 22,3.$

Für $\epsilon = 0,5$ muß $k \approx 1,17\sqrt{|Z|}$ sein, d.h. k hängt nur von $|Z|$ ab $\longrightarrow$ für Hashfunktionen ist eine Bitlänge von mind. 128 zu verwenden.

7.3 Erweiterung on Hashfunktionen

Sei $h : \mathbb{Z}_2^m \rightarrow \mathbb{Z}_2^t$ mit $m \geq t+2$. Wir konstruieren eine Hashfunktion $h^x : X \rightarrow \mathbb{Z}_2^t$ mit $X = \bigcup_{i=m}^{\infty} \mathbb{Z}_2^i$.

Für $x, y \in X$ bezeichne $|x|$ die Länge von x und $x \parallel y$ die Konkatenation von x und y.

Jedes $x \in X$ mit $|x| = n > m$ kann geschrieben werden als $x = x_1 \parallel x_2 \parallel \ldots \parallel x_k$, mit $|x_1| = \ldots = |x_{k-1}| = m-t-1$ und $|x_k| = m-t-1-d \qquad 0 \leq d \leq m-t-2$.
Also $k = \lceil \frac{n}{m-t-1} \rceil$.

Die erweiterte Hashfunktion h^* ist wie folgt definiert:

1. $FOR\ i \leftarrow 1\ TO\ k-1\ DO\ y_i \leftarrow x_i$
2. $y_k \leftarrow x_k \parallel 0^d$
3. $y_{k+1} \leftarrow$bin. Darstellung von d
4. $g_1 \leftarrow h(0^{t+1} \parallel y_1)$
5. $FOR\ i \leftarrow 1\ TO\ k\ DO\ g_{i+1} \leftarrow h(g_i \parallel 1 \parallel y_{i+1})$
6. $RETURN(g_{k+1})$

7.7. Satz *Sei* $h : \mathbb{Z}_2^m \rightarrow \mathbb{Z}_2^t$ *stark kollisionsfrei,* $m \geq t+1$. *Dann ist* h^* *ebenfalls stark kollisionsfrei.*

Beweis: Annahme: Wir hätten $x \neq x'$ mit $h^*(x) = h^*(x')$. Dann konstruieren wir eine Kollision für h in polynomieller Zeit, was zu unserer Annahme führt. Also muß h^* stark kollisionsfrei gewesen sein. Die Konstruktion: (für alle Konstruktionen ausgehend von x')

1. Fall: $|x| \not\equiv |x'| \; mod \; m-t-1$
Also $d \neq d'$ somit $y_{k+1} \neq y'_{l+1}$
$h(g_k \parallel 1 \parallel y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{l+1} = h(g'_l \parallel 1 \parallel y'_{l+1})$: Kollision!.

2. Fall $|x| \equiv |x'| \; mod \; m-t-1$.

1. $|x| = |x'| \Rightarrow k = l, y_{k+1} = y'_{l+1}$
   $h(g_k \parallel 1 \parallel y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{k+1} = h(g'_k \parallel 1 \parallel y'_{k+1})$
   Falls $g_k \neq g'_k \longrightarrow$ Kollision.
   Somit: $g_k = g'_k$; iteriere: $h(g_{k+1} \parallel 1 \parallel y_k) = g_k = g'_k = h(g'_{k-1} \parallel 1 \parallel y'_k)$ entweder Kollision oder
   $y_{k-1} = g'_{k-1}$ und $y_k = y'_k$
   usw. bis $h(0^{t+1} \parallel y_1) = g_1 = g'_1 = h(0^{t+1} \parallel y'_1)$.
   Entweder Kollision oder $y_1 = y'_1$, also $y_i = y'_i, 1 \leq i \leq k+1 \Rightarrow x = x'$ nach Konstruktion, was zur Annahme führt.
2. $|x| \neq |x'|$. Ohne Beschränkung der Allgemeinheit sei $|x'| > |x|$, also $l > k$. Der Beweis verläuft nun analog wie oben (Fall 2.1) bis $h(0^{t+i} \parallel y_1) = g_1 = g'_{l-k+1} = h(g'_{l-k} \parallel 1 \parallel y'_{l-k+1})$.
   Nun $t+1$-Bit $= 0$, $t+1$-Bit $= 1$ Kollisionen!

Die Anzahl der Iterationen ist $l = \log|x|$, also haben wir polynomielle Laufzeit. ■

#Schritte: O() = $O(log_2 x)$

7.4 Hashfunktionen von Kryptosystemenen

Idee: Verwende die Kryptosystemverschlüsselung als Teil des Hashens.

Annahme: $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_2^n$
$x = x_1 \parallel \ldots \parallel x_k, x_i \in \mathbb{Z}_2^n$.

$g_0 = IV, g_i = f(x_i, g_{i-1}), h(x) = g_k$

7.8. Beispiel

$$\begin{aligned} g_i &= e_{g_{i-1}}(x_i) \oplus x_i \qquad (\oplus g_{i-1}) \\ &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \qquad (\oplus g_{i-1}) \end{aligned}$$

7.9. Definition *p Primzahl, $q = \frac{p-1}{2}$ Primzahl, $\alpha, \beta \in \mathbb{Z}_p^*$ primitiv. Die* **Chaum-van Heijsr-Pfitzmann**-*Hashfunktion ist wie folgt definiert:*
$h : \{\mathbb{Z}_q \times \mathbb{Z}_q \to Z_p^*\}$
$(x_1, x_2) \mapsto \alpha^{x_1}\beta^{x_2}(mod\ p)$

7.10. Satz *Gegeben sei eine Kollision für diese Hashfunktion. Dann kann der Diskrete Logarithmus $log_\alpha \beta$ in Polynomzeit berechnet werden.*

Beweis: Sei $h(x_1, x_2) = h(x_3, x_4)$ mit $(x_1, x_2) \neq (x_3, x_4)$. Also
$\alpha^{x_1}\beta^{x_2} = \alpha^{x_3}\beta^{x_4}(mod\ p)$
$\Leftrightarrow \alpha^{x_1-x_3} = \beta^{x_4-x_2}(mod\ p)$.
Sei $d = ggT(x_4 - x_2, p-1) = ggT(x_4 - x_2, 2q) \in \{1, 2, q, p-1\}$.

1. Fall:

$$\begin{aligned} d = p-1 \quad &\Rightarrow \quad x_4 = x_2(mod\ q) \\ &\Rightarrow \quad \alpha^{x_1} \equiv \alpha^{x_3}(mod\ p) \\ &\Rightarrow \quad x_1 \equiv x_3(mod\ p) \\ &\longrightarrow \quad \text{Annahme} \end{aligned}$$

2. Fall: $d = q$, aber $0 \leq x_2, x_4 \leq q-1 \Rightarrow -(q-1) \leq x_4 - x_2 \leq q-1 \longrightarrow$ Annahme.

3. Fall:

$$\begin{aligned} d = 1 \quad : \quad & y := (x_4 - x_2)^{-1} mod\ (p-1) \\ & \beta \equiv \beta^{(x_4-x_2)y} \equiv \alpha^{(x_1-x_3)y}\ mod\ p \\ \rightarrow \quad & log_\alpha \beta = (x_1 - x_3)(x_4 - x_2)^{-1}\ mod\ (p-1) \end{aligned}$$

4. Fall: $d = 2 : y := (x_4 - x_2)^{-1}\ mod\ q$ (da $ggT(x_4 - x_2, q) = 1$)
also $(x_4 - x_2)y = kq + 1$ für $k \in \mathbb{Z}$.

$\beta = \beta^{(x_4-x_2)y} \equiv \beta^{kq+1} \equiv (\beta^q)^k \cdot \beta \equiv (-1)^k\beta\ mod\ p$
$\Rightarrow \alpha^{(x_1-x_3)y} \equiv \beta^{(x_4-x_2)y} \equiv (\pm 1)\beta\ mod\ p$, also $log_\alpha \beta = (x_1 - x_3)y$ ■

7.5 MD4 (1990: Rivest)

Vorbemerkung: MD4 ist unsicher! Der Algorithmus wurde von Dobbertin (BSI) geknackt. **MD5** stellt eine Verbesserung von MD4 dar (1991Rivest), wird aber ebenfalls als unsicher angesehen. **SHS (SHA)** (Secure Hash Standard, 1992-1994) basiert auf einer ähnlichen Methode, gilt jedoch als sicher.

1. Vorberechnung: $M = x \,\|\, 1 \,\|\, 0^{447-|x|\ mod\ 512} \,\|$ (beim $|x|\ mod\ 2^{64}$).
Die Länge von M ist ein Vielfaches von 512.

2. Es werden immer 16 32-Bit-Wörter gleichzeitig behandelt, dies werden mit 4 32-Bit-Registern verarbeitet. Für die je 16 Wörter werden 3 **Runden** durchgeführt.

3. Die Ausgabe ist die Konkatenation der 4 Register

Implementation: Hardwareabhängig
big-endian(SHA) $a_1a_2a_3a_4$ ein 32Bit-Wort, repräsentiert $a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$
vs. littel-endian (MS4,5): $a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$

Auf einer anderen Archtitektur vertauschen.

MD5: eine Runde mehr.

SHA: erst Expansion von 16 Worten auf 80 Worte dann 80

Expansion ausführen: $x_j \;=\; x_{j-3} \oplus x_{j-8} \oplus x_{j-14} \oplus x_{j_16} << 1 \qquad 16 \leq j \leq 79$

(Verbesserung 1994)